

AD-A259 852



Technical Report
CMU/SEI-92-TR-15
ESC-TR-92-015

2



Carnegie-Mellon University
Software Engineering Institute

DTIC
ELECTE
FEB 03 1993
S B D

Guide to CASE Adoption

Kimberly Steplen Oakes
Dennis Smith
Ed Morris

November 1992

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

98

2 2

011

4162

93-01899



6308

Carnegie Mellon University does not discriminate and Carnegie Mellon University is required not to discriminate in admission, employment or administration of its programs on the basis of race, color, national origin, sex or handicap in violation of Title VI of the Civil Rights Act of 1964, Title IX of the Educational Amendments of 1972 and Section 504 of the Rehabilitation Act of 1973 or other federal, state or local laws or executive orders.

In addition, Carnegie Mellon University does not discriminate in admission, employment or administration of its programs on the basis of religion, creed, ancestry, belief, age, veteran status, sexual orientation or in violation of federal, state or local laws, or executive orders. While the federal government does continue to exclude gays, lesbians and bisexuals from receiving ROTC scholarships or serving in the military, ROTC classes on this campus are available to all students.

Inquiries concerning application of these statements should be directed to the Provost, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, Pa. 15213, telephone (412) 268-6684 or the Vice President for Enrollment, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, Pa. 15213, telephone (412) 268-2056.

Technical Report
CMU/SEI-92-TR-15
ESC-TR-92-015
November 1992

Guide to CASE Adoption



Kimberly Stepien Oakes
Dennis Smith
Ed Morris

CASE Environments Project

Approved for public release.
Distribution unlimited.

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

This technical report was prepared for the


SEI Joint Program Office
ESC/AVS
Hanscom AFB, MA 01731

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

Review and Approval

This report has been reviewed and is approved for publication.

FOR THE COMMANDER



Thomas R. Miller, Lt Col, USAF
SEI Joint Program Office

Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By _____	
Distribution/	
Availability	
Dist	Sp

DTIC QUALITY INSPECTED 3

The Software Engineering Institute is sponsored by the U.S. Department of Defense.

This report was funded by the U.S. Department of Defense.

Copyright © 1992 by Carnegie Mellon University.

This document is available through the Defense Technical Information Center. DTIC provides access to and transfer of scientific and technical information for DoD personnel, DoD contractors and potential contractors, and other U.S. Government agency personnel and their contractors. To obtain a copy, please contact DTIC directly: Defense Technical Information Center, Attn: FDRA, Cameron Station, Alexandria, VA 22304-6145.

Copies of this document are also available through the National Technical Information Service. For information on ordering, please contact NTIS directly: National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22161.

Copies of this document are also available from Research Access, Inc., 3400 Forbes Avenue, Suite 302, Pittsburgh, PA 15213.

Use of any trademarks in this report is not intended in any way to infringe on the rights of the trademark holder.

Table of Contents

1	Introduction	1
1.1	Purpose of This Guide	1
1.2	Organization of This Guide	2
2	Executive Overview	3
2.1	Origins of "CASE"	3
2.2	Impact of CASE Tools	3
2.3	Major CASE Issues	5
2.4	Making an Informed Decision	5
2.4.1	Awareness and Commitment	7
2.4.2	Selection	8
2.4.3	Trial (Implementation)	8
2.4.4	Implementation Strategy	9
2.4.5	Routinization	9
3	Common CASE Questions and Answers	11
3.1	Should I purchase CASE tools? Which ones?	11
3.2	Will new tools help me improve my software process?	12
3.3	How does the life-cycle model affect my tools?	13
3.4	Which methodology should I choose?	13
3.5	Why do I need a CASE strategy?	14
3.6	How do new tools fit within my current environment?	14
3.7	What are the long-term implications of a tool decision?	15
3.8	How are tools integrated?	16
3.8.1	Platform Integration	17
3.8.2	Presentation Integration	17
3.8.3	Control Integration	17
3.8.4	Data Integration	18
3.8.5	Process Integration	19
3.9	Can I integrate tools myself?	19
3.10	Can tools handle large applications?	20
3.11	How do I determine the "openness" of a tool?	21
3.12	Must all tools be "open"?	22
3.13	Will vendors customize a tool for me?	23
3.14	What do vendor integration claims mean?	24
3.15	What do CASE tools cost?	24
3.15.1	Principal Cost Drivers	25
3.15.2	Start-Up Costs	25
3.15.3	Ongoing Costs	26
3.16	What determines the impact of a tool?	26
3.17	How do I maximize my return on investment?	27

3.18 What are major non-technical tool considerations?	28
3.19 Should I attempt to build my own tools?	29
3.20 What new tool technology is on the horizon?	29
4 Toward a CASE Adoption Strategy	31
4.1 Assess the Organization	34
4.1.1 Evaluating the Organization's Personnel	34
4.1.2 Evaluating the Organization's Technology	34
4.1.3 Evaluating the Organization's Process	35
4.1.4 Evaluating the Organization's Projects	36
4.1.5 Evaluating the Organization's Politics and Perceived Needs	36
4.2 Assess Available Technology	36
4.2.1 Tool Analysis	37
4.2.2 Vendor Analysis	38
4.2.3 User Experiences Analysis	39
4.3 Assess the Suitability of the Technology	39
4.3.1 Evaluating and Selecting the Tools	39
4.4 Pilot the Tool	41
4.4.1 Experiences and Lessons Learned	41
4.5 Transition the Tool	42
4.5.1 Culture Change	42
4.5.2 Training	43
4.5.3 Project Standards	44
4.5.4 Evaluating the Effectiveness of CASE Tools	44
4.6 Institutionalize Tool Use	45
5 Conclusion	47
References	49
Appendix A Acronyms Used in This Guide	51
Appendix B CASE Adoption Resources	53

List of Figures

Figure 2-1	CASE Adoption Stages	7
Figure 3-1	Start-Up CASE Investment	25
Figure 3-2	Ongoing CASE Costs	26
Figure 3-3	Factors That Influence CASE Impact	27
Figure 4-1	CASE Adoption Strategy	33

List of Tables

Table B-1:	U.S. Government CASE Information Sources	54
Table B-2:	CASE Industry Specific Reports/Directories	55
Table B-3:	CASE Industry Specific Magazine-Based Buyer's Guides	56
Table B-4:	General Software Industry Reports/Directories	56
Table B-5:	Consulting Groups/Conferences	57
Table B-6:	CASE Industry Newsletters	57
Table B-7:	CASE Trade Shows	58
Table B-8:	CASE User Groups	58

Guide to CASE Adoption

Abstract: In an attempt to address the productivity and quality problems afflicting the software industry, many organizations are turning toward computer-aided software engineering (CASE) technology as a potential solution. Unfortunately, the inflated claims of vendors and unreasonable expectations of new users have led to many failed CASE adoption efforts. This guide answers questions organizations may have concerning CASE technology, and provides a strategy for the adoption of CASE tools into an organization.

1 Introduction

The software problem is not new, nor is it about to be solved. The SEI software capacity study [Siegel, Stewman, Konda, Larkey, & Wagner, 1990] has documented the critical software needs facing the United States military over the next several decades. The software demands of the commercial sector are similar. Indicators of the existing software capacity problems include:

- Insufficient resources to develop software that is currently projected, leading to a search for greater productivity.
- Problems in existing software, resulting in a search for better quality of delivered software and an increased demand for assurance that requirements are clearly stated and implemented.
- Large costs in the maintenance of existing software, resulting in a need for tools to restructure code, generate clear documentation, and manage multiple configurations of software.

These problems have encouraged a search for new methods and tools for developing software more effectively. Computer Aided Software Engineering (CASE) tools represent a promising technology which may eventually allow us to address some of our deficiencies in building software.

Unfortunately, when CASE tools were first introduced, a number of inflated claims led people to believe that CASE tools would be an immediate and primary solution to the many problems in developing and maintaining software. While such claims continue to be common, the cumulative experience of CASE tool users is sobering. However, this experience may provide us with the data to sort out the many (and often conflicting) claims.

1.1 Purpose of This Guide

This guide is intended to help project managers make informed decisions about various CASE claims, and additionally to support them in decisions concerning when and how to introduce

CASE technology into an organization. The guide identifies some major areas of concern, provides guidance for addressing these issues, and identifies sources for additional information.

The views presented in this guide are derived from interviews with managers in organizations that have adopted CASE technology, from workshops sponsored by the SEI, from our own experience, from the work of others at the SEI, and from a review of the published literature concerning CASE technology. The ideas and concepts presented are derived from a number of disciplines including software engineering, management science, and the social sciences.

The intent is that this is a "living" guide that evolves over time due to changing technology, feedback from users of this guide, and as our understanding and perspectives on the issues change. Periodic republication of this guide is planned. As a result, feedback, comments, and suggestions for improvements to this guide are most welcome.

1.2 Organization of This Guide

This guide provides information at several levels of detail for individuals with varying information needs. It is not expected that everyone will need or want to read the entire guide. For high level managers, Section 2 offers an executive overview of the major issues covered. A more in-depth understanding of CASE issues, and answers to commonly asked questions concerning CASE technology, are provided in Section 3. Finally, Section 4 draws from the work of experts in the fields of technology transition and CASE tools to define a strategy for adopting CASE technology.

2 Executive Overview

Because of the nature of the software problem, many organizations are attempting to increase the productivity and quality of their software development efforts. Some have turned to CASE tools as an aid in developing better software. Initially, CASE tools were hailed as a panacea for software development problems, with the assumption that the use of tools would by themselves produce dramatic increases in productivity. Recently, there has been a recognition that tools represent only one factor in the improvement of software development efforts.

This guide addresses many of the concerns of managers in the formulation of a CASE strategy. It identifies the different types of CASE tools and the role of tools within the context of the software development process, the methods used within an organization, the hardware and software environment, and the personnel who will use and maintain the system. A number of technical issues are highlighted including tool integration, data management, performance, maintainability, and standardization. In addition, non-technical issues of relevance to managers are considered including the tool selection, the adoption process, and culture change. Each of these issues represents an important consideration in the formulation of a CASE strategy.

2.1 Origins of "CASE"

The term Computer Aided Software Engineering was first applied to tools which provided support for the *analysis and design phases of the software development cycle*. Many of the early tools automated structured methods that had been available but were infrequently used due in part to the lack of automated support.

Later, there emerged other categories of tools providing automated support for software engineering. The acronym CASE was applied to the wider range of tools. Currently the vision of CASE is that of an interrelated set of tools which support all aspects of the software development process. These include tools which support specific phases of the life cycle, such as analysis and design tools, code generators, and testing tools; and tools which provide functionality across the life cycle, such as project management tools, configuration management tools, and documentation tools.

We use the term CASE here to refer to this wider range of interrelated tools that support the software engineering process. Where data is specific to a particular type of tool, we have identified that type.

2.2 Impact of CASE Tools

It is imprecise to make blanket statements about the benefits of CASE technology due to the diverse nature of CASE tools. Some tools, such as configuration management tools and documentation tools, are generally accepted mechanisms for improving the manner in which software is developed. More controversial are the benefits of other tools, such as the many

analysis and design tools, reverse engineering, or code generation tools which are commercially available. Among the major problems with careful measurement of the impact of any type of CASE tool are:

- The wide variation in quality and value within a single type of tool.
- The relatively short time that many types of CASE tools have been in use in organizations.
- The wide difference in the adoption practices of various organizations.
- The general lack of detailed metric data for previous and current projects.
- The wide range of project domains.
- The confounding impact of changes to methods and processes that are often associated with the adoption of CASE tools.
- The potential bias of organizations reporting CASE gains or losses.

Illustrative of the problem with measuring CASE impact are the varying experiences of organizations that have purchased and used CASE analysis and design tools. A review of the literature indicates that many industry analysts document productivity gains ranging from 10% to 30% resulting from CASE analysis and design tool usage, with similar modest gains in software quality and documentation. However, negative impact has also been experienced, although documented publicly less frequently. These figures are derived primarily from anecdotal data, and therefore suffer from potential "biases" either toward or against the tool.

A review of the literature also indicates that the data gathered and effects of CASE tools experienced varies widely among organizations. Measurement differences appear both in the manner in which productivity and quality are measured and in the quality of the data gathered. Organizations have experienced variances in the impact of CASE tools depending on project size, customer involvement, and user sophistication. In addition, some CASE analysts expect that true gain is only realized after 1-2 years of experience, while others suggest that the impact may only become evident during the maintenance phase of the software lifecycle, where improved software design reportedly attributable to CASE technology leads to lower maintenance costs.

The most consistent benefits cited in the CASE literature are improved communications and documentation. Communications appears to be enhanced both between the project engineers and the customer and among engineers working on a project. The improvement is often attributed to more accurate, consistent, and understandable representations of a system which are potentially possible with CASE tools. The experienced improvement in documentation relates to this greater consistency and accuracy for specifying the system and to the direct generation of portions of the documentation by the CASE tools. Such documentation capabilities may be particularly important in the government sector where documentation costs can comprise up to 40% of project cost. However, experienced users of CASE tools reject the extravagant documentation claims made by some CASE vendors and suggest that large portions of the documentation process remain manual.

2.3 Major CASE Issues

Clearly, the decision to invest in CASE technology is not easy to make. An organization must make informed assessments about the value of a wide variety of individual tools based on inconclusive data. Determining the potential value of a tool is made even more difficult to measure in isolation, since an increasing number of CASE researchers and users have reached the conclusion that the greatest value of CASE tools will only be achieved when used in concert with other tools (i.e., integrated).

Organizations that are attempting to reach consensus on the purchase of CASE tools must address a number of issues. The positions adopted by an organization can determine whether a CASE tool will be ultimately successful. Among the more vexing issues are:

- **Investment of Resources.** The cost of adopting CASE technology.
- **Current Processes and Methods.** The frequently inexact match between the processes and methods supported by CASE tools and those utilized within the organization.
- **Support Mechanisms.** The extensive support systems necessary for CASE tools.
- **Tool Scalability.** The somewhat limited capabilities of CASE tools to deal with very large systems.
- **Assessment of Real Value.** The difficulty in determining the actual value of CASE tools when faced with sometimes inflated claims from vendors.
- **Standards Selection.** Selecting among the sometimes competing and conflicting standards supported by CASE tools.
- **Adoption Complexity.** The complexity of the tool adoption process.

These issues and others are addressed in Section 3.

2.4 Making an Informed Decision

Although empirical data to objectively analyze the impact of CASE tools on software development is limited, a survey of the (primarily) anecdotal data available indicates a consistent clustering of benefits attributed to CASE technology. A list of these commonly cited benefits includes:

- variable productivity gains
- modest quality gains
- improved documentation
- enhanced project communications
- enforcement of project methodology and standards

However, any decision to bring a CASE tool into an organization should be made with an awareness of both short-term and long-term implications of tool adoption. Over the short term, organizations adopting CASE tools should be willing to accept:

- a potential decrease in productivity
- dissatisfaction on the part of employees adopting the new technology
- changes to process and methods
- potentially extensive training
- significant expense

Over the longer term, CASE organizations must address:

- Long term maintenance costs of CASE tools (potentially for the life cycle of systems developed with the tool).
- Frequent releases of new technology.
- The potential that development of environment/tool frameworks such as PCTE will lead to major restructuring of tools.
- Continual costs for training new staff and upgrading the skills of existing staff.

The success or failure of a CASE adoption effort depends largely on the ability of an organization to manage short- and long-term costs. Organizations which have addressed these problems in a well conceived adoption process stand the best chance of success. This approach contrasts with others which focus primarily on the mechanics of choosing a particular tool.

The SEI is developing an outline of a CASE adoption process which addresses these and other concerns. [Tornatzky & Fleischer, 1990] outlined the stages of incorporating a new tool or practice as a variant of a pattern: awareness-problems, matching-selection, adoption-commitment, implementation, and routinization. We have used the stages of Tornatzky and Fleischer as a foundation, and modified them specifically for CASE technology. The resulting model postulates six stages for the CASE adoption process: awareness, commitment, selection, trial implementation, implementation strategy, and routinization. These stages, illustrated in Figure 2-1, represent a cycle where each stage provides the input for the next. Depending on the maturity of an organization prior to the adoption effort, some of the preliminary stages may have already been completed.

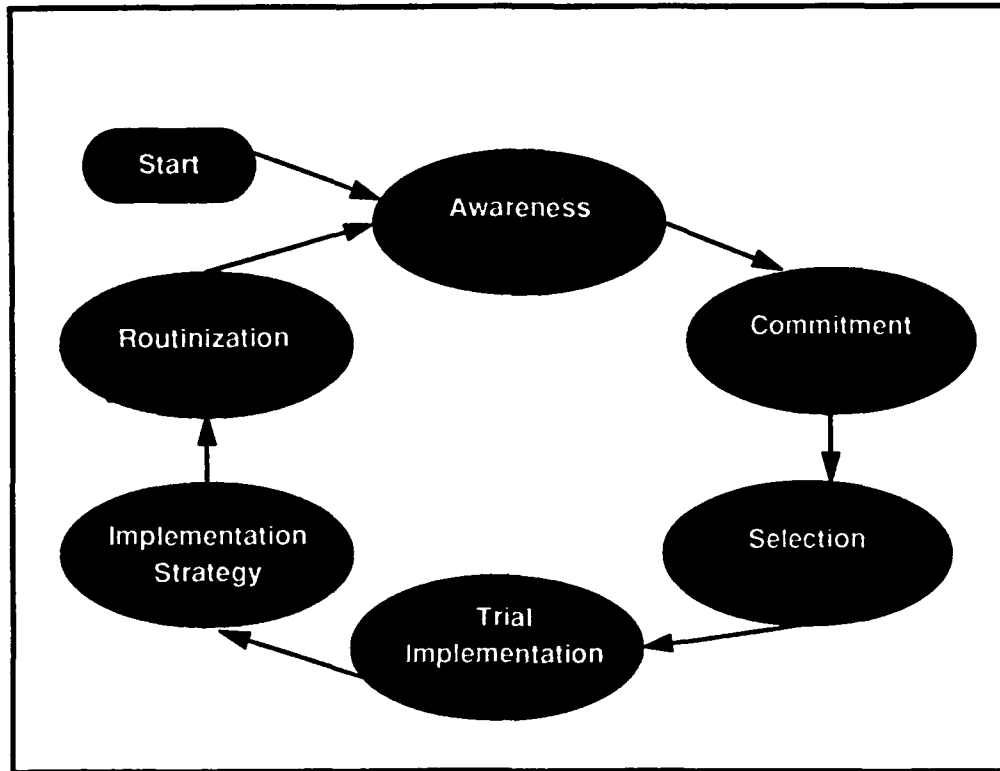


Figure 2-1 CASE Adoption Stages

2.4.1 Awareness and Commitment

Most organizations perform a preliminary search for information about CASE tools long before they have made any commitment to adopt CASE technology. During this awareness phase, it is useful to become versed in CASE literature, attend workshops and seminars, and solicit information from other organizations which have adopted CASE technology. Experience at SEI CASE workshops suggests that perhaps 50% of the attendees are involved primarily to become aware of strengths and weaknesses of CASE technology.

The commitment stage consists of the decision process to adopt CASE tools. Commitments from both management and those who will be utilizing, or otherwise impacted by the tool are essential. A common mistake in this phase is to diminish the importance of commitment from the managers, engineers and support personnel whose daily activities will be affected by the incorporation of a new tool technology.

Ironically, because management is reluctant to disrupt current practice to introduce the organizational changes required for the successful implementation of CASE, it often represents the greatest barrier to increased software productivity [Forte, 1988]. The easiest way to convince management to dedicate resources for change is to prove that substantial gains in productivity and quality can be achieved by implementing CASE technology. Unfortunately, this evidence is not readily available. Other methods of increasing the commitment of management include

developing precise definitions about the change and the methods of measuring it, providing education about the need for tool support, and establishing common goals and objectives for the project.

2.4.2 Selection

While CASE tools can be (and often have been) purchased in isolation, a more effective approach to the selection process requires the development of a tool strategy. The strategy should address both the short- and long-term needs of the organization, based on overall process and technology improvement directions.

Based on the needs identified in a tool strategy, the choice of an individual tool can begin. An appropriate tool selection approach includes:

- narrowing down the list of available tool options to a small number of tools,
- determining how the new tool will interact with other tools in the environment,
- analyzing candidate tools according to both technical and non-technical criteria, and
- testing the candidate tools.

2.4.3 Trial (Implementation)

Once a tool has been tentatively selected, it is important to try out the tool on a pilot project. Many organizations skip this step, as it entails devoting significant resources, including personnel, time, and money. However, only a pilot project, carried out under actual conditions, will determine what the tool offers, how it works, how effectively it performs its task, and its shortcomings. These issues are simply too complex to make an informed decision without a trial evaluation. Vendor demonstrations can be helpful, but are not sufficient for making informed decisions. Although most vendors will make a copy at little or no cost, organizations must ensure that the in-house evaluation is realistic. Tools best demonstrate their true capabilities with real data and not in the contrived environment of a vendor's tutorial.

If management sees that the tool aids the development process during a pilot project, they may support its continued use. When pilot projects fail to show performance or quality gains immediately, management may grow discouraged. However, even a successful pilot project can increase risk by raising management expectations that the (unrealistically) positive results will transfer directly to the larger organization. For these reasons, management must be clearly informed and hold realistic expectations for success for the pilot project and for transition of the new technology.

During the hands-on testing period, it will be important to perform an objective analysis through a full development cycle, with realistic simulations of database size and multiple users. This type of hands-on testing will give a better idea of the specific functions provided by individual tools and how various tools will work together.

Proper implementation of CASE tools can lead to better quality software. Finding the best way to use these tools however, is a difficult process. Starting with small pilot projects can make implementing large-scale efforts much easier later on. This early work needs adequate staffing and appropriate leadership, along with the resources necessary to develop standards for staff training and tool operation. According to vendors and users alike, at least one successful pilot project demonstrating CASE capabilities is the best way to secure organizational commitment [Feuche, 1989]. Also, such a pilot project can highlight tool capabilities, lead to the development of tool standards, and provide a head start on the construction of reusable component libraries.

2.4.4 Implementation Strategy

The primary challenge for migrating a CASE tool into general use involves integrating the new tool into the existing environment while minimizing (or at least managing) the organizational disruption. The disruption of a new tool can effect many elements of the environment, including the personnel and the existing process and methods.

The problems of implementing a new tool are similar to those caused by the adoption of many other new technologies. Effected personnel are often ambivalent and occasionally negative about the change, and may need to be reassured about their role in the organization. The characteristics of the tool may require processes and methods be modified. The tool itself may require customization to support organizational needs.

Poor implementation strategies often lead to the shelving of the tool. There are many stories telling of hundreds of copies of a tool gathering dust. It is important to recognize that the task of implementing a tool is not completed by careful selection of the tool and the subsequent demonstration of tool capabilities on a pilot project. It also should be recognized that mandating the use of a tool can be counter-productive. An good implementation strategy is one that encourages tool use by rewarding individuals and projects that "make it work."

2.4.5 Routinization

It is frequently stated that software maintenance is the longest and most expensive phase of the software lifecycle. For a successful and cost effective maintenance process, an infrastructure should be built to facilitate incorporation of periodic upgrades, provide training, and support corporate decisions concerning new directions. Routinization of tool use marks the beginning of the maintenance phase for a CASE tool. A number of efforts to adopt tools have failed because of an inability to incorporate the tool into the day to day activities and planning of the organization.

Major challenges of a CASE tool routine include the indoctrination of new employees into the system and the continual upgrading of skills of existing employees. A common mistake is to provide initial training for a group of early users, followed by only minimal ongoing training. Unfortunately, it is the larger group of users who are not CASE tool "pioneers" who potentially require more training.

Another common mistake is to underestimate the resources necessary to support continual use of complex CASE tools. One factor that adds to resource demands is the complexity of the tool. Many CASE tools require experienced personnel capable of managing the tool databases and responding to problems. A second factor involves the frequent release schedules of CASE tools. While many of the tools have matured to the point where data incompatibility problems between versions are minimized, dependencies on the rest of the computing environment (such as the operating system version) can cause configuration nightmares.

Section 4 of this guide contains a proposed CASE adoption strategy based on the CASE adoption model, and on the technology transition work of others (including [Tornatzky & Fleischer, 1990] and [Przybylinski, Fowler, & Maher, 1991]). Guidance is provided on assessing an organizations readiness to adopt CASE technology and improving the selection, implementation, and routinization processes.

3 Common CASE Questions and Answers

Members of the CASE Technology Project at the Software Engineering Institute are frequently approached by organizations considering the adoption of CASE tools. These organizations have very similar questions concerning CASE tools and CASE tool adoption. This is not surprising, because most organizations are struggling with similar difficulties in generating and maintaining software.

Subsequent sections of this guide will be devoted to identifying and answering some of these frequently asked questions about CASE technology. The views presented in this guide are derived from interviews with managers in organizations that have adopted CASE technology, from workshops sponsored by the SEI, from our own experience, from the work of others at the SEI, and from a review of the published literature concerning CASE technology.

3.1 Should I purchase CASE tools? Which ones?

Some organizations have benefited from CASE tool installation, while others have gained little from the considerable expense. Because most tool vendors have both satisfied and unsatisfied customers, the diverse range of values returned on CASE tool investment is not due solely to individual differences in tools,

Instead, some of the differences depend on the characteristics of the adopting organization. In order to predict the value of a CASE tool, these characteristics need careful analysis:

- **Personnel Factors.** These include tangible factors such as the willingness of personnel to change. [Andrews, 1989] suggests that junior engineers were more willing to adopt a new technology, while experienced engineers were more likely to resist the introduction of CASE tools and new development methodologies. In general, a policy which offers few surprises is often best.
- **Computing Resources.** Within an organization, the computing resources can influence the value derived from a tool purchase. Buying a tool may require the purchase of new hardware as well. Conversely, insufficient computer resources (e.g., inadequate CPU, disk and network capacity) can severely affect the performance of CASE tools. Also, some CASE tools require substantial support from personnel to manage access lists and licenses, and to maintain contact with the vendor.
- **Organizational Sophistication.** A more sophisticated organization requires greater formal accounting of the software process. Thus, they will likely benefit from the purchase of CASE tools. Likewise, the stability of their process puts them in a better position to recoup the initial expense of CASE tools. A reasonable starting point for determining an organization's sophistication is the SEI capability maturity model (CMM) for software [Paulk et al., 1991].

- **Organizational Needs.** Both perceived and actual needs can influence considerably the types of tools that are appropriate, the adoption process necessary, and the kinds of standards that are required. It is critical to perform a realistic analysis of the needs of your organization before investing in any CASE tool. A common mistake of managers is to misunderstand where the major effort is spent during software development in their organization. A rule of thumb for long-lived DoD applications is that significantly more money will be spent on maintenance efforts than on original development. Therefore, the most cost-effective tool over the long run is likely to be the one that improves the maintenance process to the greatest degree. Another rule of thumb for the development portion of the life cycle for DoD applications is that the greatest single cost during development is for documentation efforts [Jones, 1990]. Therefore, a tool that simplifies documentation may be appropriate for many organizations.
- **Project Size and Domain.** Both of these factors can influence greatly the selection and use of tools. For example, a large project will need to pay careful attention to such issues as cooperative processing, database size, and project management. A hard real-time project will be concerned with issues such as simulation and scheduling, while an MIS project will require strong database capabilities.
- **Organizational Expectations.** The appropriate tool will depend on what your organization expects from it. If you expect immediate productivity or quality improvement on the next piece of software developed, then it may be inadvisable to purchase an analysis and design tool requiring significant training and a long term commitment on a methodology. On the other hand, if such a purchase is part of a carefully thought out plan for long-term improvement in methods and process, then the tool may be a worthwhile investment. Note that other types of CASE tools may require less training, and have a shorter period for ROI. These tools may address immediate needs more effectively.

3.2 Will new tools help me improve my software process?

Increased competition and, in some cases, SEI assessments are causing many organizations to reconsider how effectively they develop software. It is natural for them to look to additional tool support as one method of improving their software process. In fact, the SEI capability maturity model lists the provision of appropriate resources (including tool support) as a key process area in maturity levels 2-5, and in a number of places provides examples of tools.

A key, but often overlooked point, however, is that the tools used by a software organization must reflect the software methods and practices in place within the organization. [Smith & Oman, 1990] advise that an overall tool strategy for an organization or project should consider the process and methods in place, determine the functions of the life cycle that need to be supported, and account for the environmental infrastructure that will let tools work together.

For a tool to be successful in a large-scale development project, it must support a process that is well entrenched rather than existing outside the bounds of the process. Those tools that offer a useful service, but do not fit smoothly into the existing process, are not likely to be suc-

cessfully used on large projects. Likewise, tools can present additional problems due to the data transfer incompatibilities between other tools in different development stages requiring that data be reentered manually.

Organizations at different levels of process maturity are likely to have different needs for tools and to use tools differently. A reasonable first step is to analyze the process maturity level, develop a plan for improvement, and incorporate the tool strategy into the improvement plan.

3.3 How does the life-cycle model affect my tools?

What model of software development an organization accepts will influence its practices. During the 1970s and much of the 1980s, the waterfall model was the accepted model for the software development process. The software maintenance process was often viewed as a smaller scale analog of the development process. It is not surprising, then, to find the major features of the waterfall model embedded in government standards such as DOD-STD-2167A, in the methodologies supported by commonly available tools, and in software environments based on these standards and tools.

Recently, additional software development models have been suggested. These include the spiral model [Boehm, 1988] and various models based on rapid prototyping and reuse of existing software components. It is unclear whether any of these new models will supersede the waterfall model as the accepted standard. In fact, it is most likely that the software community is entering a period where multiple models of the software development cycle are common.

Different development models can place a premium on different tools. For example, a rapid prototyping model of development may require tools to rapidly generate user interfaces. Other influences of the development model on software tool support may be less obvious. For example, the use of a cyclic model may require rapid and frequent transitions between life-cycle phases, and therefore require a tightly integrated tool set to minimize life-cycle disruption. On the other hand, a traditional waterfall model may be adequately supported by a less tightly integrated tool set, or even by manual conversion of data from phase to phase.

3.4 Which methodology should I choose?

Current CASE analysis and design tools primarily support structured methods like those proposed by Yourdon. In addition to these well-accepted methods, new object-oriented, rapid prototyping, and reuse-oriented methodologies are being introduced. There is debate in the software engineering community concerning the effectiveness of the various methods, particularly for large systems. Proponents of object-oriented methodologies believe that these newer methods offer better overall system design, facilitating reuse of software components. Proponents of rapid prototyping methodologies argue that they are best able to establish user requirements and reduce the risk of development. Proponents of established structured methods point to a variety of successful systems developed with their preferred methodology.

Interestingly, there is little or no empirical data to suggest that one specific method will lead to greater success. [Wood & Wood, 1990] found little to support the choice of one structured method over others. In addition, it is unclear whether the use of newer object-oriented, prototyping-oriented, or reuse-oriented methods will enhance the chances for success. Successful and unsuccessful projects have been attempted with all of the contending methodologies.

A reasonable start to identifying an appropriate tool is to first analyze your current method. If you are satisfied with your current method, there is little reason to change. If you are unsatisfied with your current method, you may be best served by investigating which methods have proven successful for your particular type of software. Once you have identified an appropriate method based on the experience of others, you should first introduce the method to your organization prior to making an actual tool purchase.

3.5 Why do I need a CASE strategy?

A tool strategy is needed to determine the relationship of a set of tools to the processes and methods of your organization, and develop an overall plan for the adoption of the tools. The strategy outlines the improvements required in the current software development process and how these improvements will be made.

In accordance with the "Awareness and Commitment" stage of the CASE adoption process (outlined in Section 2), a CASE strategy has a number of basic (or "front-end") parts, including:

- analysis of software engineering process
- analysis of methods for software development
- analysis of model for software development
- development of overall improvement plan
- analysis of current environment and tools
- development of tool strategy

Implementing tools may be only one part of a strategy. For example, a strategy centered around improving the data management of an organization may not need to incorporate tools since the bulk of the work is primarily manual. A CASE tool could certainly help enforce the data standards, but implementing a CASE tool solely for this purpose would probably not be cost effective.

Details of the development of an improvement strategy are beyond the scope of this guide, but can be found in [Fowler & Rifkin, 1990].

3.6 How do new tools fit within my current environment?

Some CASE tool vendors advertise the openness of their tool offering and imply that the tool can be easily integrated into an existing development environment. Many vendors are in fact working hard to improve the external accessibility of tool data. Even if our most optimistic ex-

expectations for open architectures were exceeded, however, it is unlikely that tools would fit into an existing environment in a "plug compatible" manner.

Often, because of its characteristics, a new tool can be difficult to fit into an organization's existing environment. Even those tools that are relatively open tend to perform their functions in an egotistic manner, as if they are the center of the tool "universe." In general, tools tend to provide unique functionality rather than relying on services provided by existing environment functions. For example, many CASE tools provide a configuration management (CM) function that does not utilize, and in some cases conflicts with the functions provided by CM systems. Similarly, instead of using existing commercial data management systems, most CASE tools provide a customized system.

This "home grown" approach to tool functionality was adopted by CASE vendors for three primary reasons: it provided complete control over the technology offered by the tool, it allowed tuning of the system for best performance, and it allowed vendors to provide a turn key system with few dependencies on other environment software. Unintentionally, the end result has been to make tools more difficult to integrate into user environments.

Perhaps a larger part of the difficulty of integrating a new tool into an existing environment, and as well as a reason why tools may never be completely "plug compatible," is the relationships between tools and the existing process, methods, and personnel. It is possible (in fact likely) that some reworking of the process and methods, as well as retraining of staff will be necessary. It has been suggested that the more tightly integrated the tools of the existing environment, the more work will be necessary to replace a tool. It is also likely that the more highly defined your process, the more tool adaptation that will be necessary. Even where tools provide integrated capabilities to perform a major step in an organization's development process (such as the generation of DOD-STD-2167A documentation), experience suggests that great effort must be expended to make the product usable due to limitations of the integration, as well as variance in the requirements of the organization.

A useful step in ameliorating the problems of integrating a new tool into an existing environment is to develop a specific action plan which addresses integration with other tools, processes, methods, and people as part of a tool strategy. The action plan should address the building of bridges between the new tool and the existing components of the environment (tools, processes, methods, and people).

3.7 What are the long-term implications of a tool decision?

It has been argued in the CASE literature that the primary benefits of CASE tools come not from the savings during initial development, but rather from savings that become apparent during maintenance [Andrews, 1989]. Thus, it can be expected that a tool that offers a large return on investment will be one that lives a long life. The consequences for the long life expectancy and cost of a CASE tool can be profound. Consider the following scenarios:

- You are buying more than a tool. You are also buying the future of a company. Because of the significant costs for the tool, plus the tremendous effort invested in making it work for the organization, you will be unlikely to convince management to switch to another tool. You can expect your original commitment, if it becomes institutionalized, to be maintained over a number of years.
- Your tool choice will exert profound influence over subsequent tool and environment decisions. Subsequent choices will be practically limited to those vendors that maintain strong relationships with your CASE tool vendor.
- Your staffing needs for training and system support will increase significantly. Among the most expensive aspects of tool adoption is the development and retention of in-house tool experts who can cope with the many tool support problems expected. Some organizations using sophisticated CASE tools have found that the level of expertise and support necessary is closer to that required for an operating system than to that required for a word processing program.
- You will need to cultivate continued support from high level management. Many organizations have chosen a CASE tool, used it with moderate success on a few projects, and then abandoned the tool due to a lack of long-term commitment to make it work in the organization's environment. Successful implementors of CASE tools often have strong advocates at high levels of management who, when faced with a project manager experiencing tool-related delays, provide appropriate support for the project and honestly address the delays, but also demand that the tool be made to work.
- You must gain control over tool and environment configurations such that you can rapidly reconfigure the tool environment of a software component. This is made difficult by the numbers and frequency of tool releases, and the developing interdependence among various versions of tools.
- You should track emerging tool and environment technology, and emerging standards, and relay your thoughts and concerns to the vendor in the hope of influencing future tool directions. One of the best ways to do this is through participation in an active and vocal user's group.

A software organization that wishes to use sophisticated tools must obviously develop a strategy to manage the change in tools over the life cycle of the affected software products. Such change is common and hardly ever comfortable, but far less painful if planned.

3.8 How are tools integrated?

While this report is not intended as a guidebook for organizations that wish to integrate tools, a background in the types and techniques of tool integration may be useful for those purchasing tools as well. An analysis of the various levels of integration can be found in [Brown & McDermid, 1991], while a more complete explanation of the types of integration is available in [Thomas & Nejme, 1992].

[Wasserman, 1990] identified five types of integration:

1. Platform Integration
2. Presentation Integration
3. Control Integration
4. Data Integration
5. Process Integration

3.8.1 Platform Integration

Platform integration refers to incorporation of a tool with a common set of services provided by the computing environment. For example, the UNIX environment provides a form of platform integration for tools. In some respects, this is the least interesting form of integration because it does not deal directly with tool-to-tool integration.

3.8.2 Presentation Integration

Presentation integration refers to the provision of a consistent user interface across tools. Such consistency can greatly simplify the use of a tool set. In addition, the time and cost of comprehensive training and support is reduced. Standardized user interfaces can ultimately lead to greater tool choice and flexibility for users.

Organizations have long tried to achieve a form of presentation integration by developing user interface standards for internal tools and by providing graphical user interface "wrappers" around (primarily command line based) external tools.

More recently the X Window System, in conjunction with the Motif look-and-feel format, has achieved broad support. It is important to recognize, however, that the X Window System/Motif combination does not provide complete presentation integration. Vendors remain free to use unique vocabulary and menu contents.

3.8.3 Control Integration

Control integration refers to the ability of tools to inform other tools of their actions and to request actions by other tools. A very rudimentary form of control integration is represented by command line invocation of one tool by another. Unfortunately, command line invocation is inadequate to provide the level of integration required by users. Users (justifiably) demand that integration occur at the level of the actions which occur in individual tools. Thus, at the moment of a design action within a CASE analysis and design tool, notice or access to other tools affected by the change would ideally be immediate.

Organizations have long achieved a rudimentary form of control integration by using mechanisms such as UNIX shell scripts to invoke tools in order to achieve an ordering of tool functioning. An increasing number of tools are incorporating more sophisticated mechanisms such as programmatic interfaces that provide access for finer-grained control over tools. Unfortu-

nately, the use of these interfaces often leads to point-to-point-integrations of individual tools, since there is no universal standard for the form and function of programmatic interfaces. Such point-to-point integrations are expensive both to create and maintain, and effectively limit the user's flexibility when replacing a tool, since competing tools are likely to provide different interfaces.

A more complex but promising mechanism includes a monitor that receives tool notifications or requests and subsequently sends appropriate notifications and requests to other tools in the environment. Hewlett-Packard's SoftBench is currently the most prominent example of this technology, although other vendors such as Digital Equipment are preparing to offer similar products. The technique requires that the monitor maintain an overview of both the other tools in the environment and the process that is to be implemented, but it offers the advantage of centralizing control integration processing. Unfortunately, tool vendors have yet to agree on the events involved in the sending or receiving of a message, however there is industry interest in generating such standards.

3.8.4 Data Integration

Data integration refers to the transfer of information between tools, and the establishment of relationships between data utilized by different tools. One common method of data integration requires that individual tools agree on specific interchange formats or interfaces. This approach is relatively simple to implement and widely applicable to many types of tools. Perhaps the most common of such interchange formats is represented by ASCII files. More elaborate interchange standards, such as CDIF (CASE Data Interchange Format) [Chappell, Downes, & Tully, 1989], are also supported by a number of tool vendors. Such methods, however, provide only for the exchange of data and are not effective at establishing links between data maintained by different tools, or at maintaining the semantic context of data.

A second approach to data integration has been the development of filters which extract portions of data from individual tools and store this data into a secondary database for processing by other tools. This approach has been commonly used to extract data from tools, organize the data along some schema, store it in a central database, and then use the data to generate reports and documents. This approach allows the generation of arbitrary relationships between tool data but, like the previous approach, results in the maintenance of point-to-point integrations (between filters and tools), as well as duplication of data in the individual tools and central database.

A third method for achieving data integration involves the development of a shared repository in which a variety of tools store information. A fully functioning repository would provide the capability of maintaining a core semantic content of objects together with tool-specific views, and because of the common dictionary would permit several tools to work together. Repositories (such as PCTE 1.5, and later ECMA PCTE) [Thomas, 1989] are becoming available, and have been discussed widely but, in our opinion, are still several years from maturity.

3.8.5 Process Integration

Process integration refers to the automation of the sequence of activities in support of an organization's defined process for the software life cycle. To achieve a high level of process integration, the mechanisms of presentation, control, and data integration are used. In effect, process integration is the end toward which the other forms of integration provide a means.

While process integration has been the overall goal of many integration attempts, little is known about the characteristics and parameters of quality process integration. Ideally, a well integrated process would support an organization's activities without mandating a single routine. The well integrated process would insure that milestones and standards are met, but provide for flexibility in the meeting of those standards and milestones.

Commercial organizations involved in the development of integrated environments must address this simultaneous need for structure and flexibility. They must develop an environment which provides an adequate degree of process integration, while at the same time allowing adequate flexibility to support a wide customer base. Fortunately, after initially devoting most of their time to the technical aspects of the solution, many environment builders now recognize the difficulty in defining and supporting a set of domain specific processes, and are pursuing solutions that provide balanced presentation, control, and data integration.

3.9 Can I integrate tools myself?

While it is possible to perform limited integrations of CASE tools yourself, many of the largest corporate-directed integration efforts have proven to be technically risky and/or too costly, and were later scaled down or abandoned. A number of problems are evident with corporate-directed integrations, among them:

- The level of integration that can be achieved without modification of individual tools is limited. While many vendors claim to offer "open" access to their tool, this is often limited to data access routines. Unfortunately, semantic information is often embedded in the tool functioning, and is therefore unavailable for integration.
- Tool vendors are not likely to perform significant modifications in support of a single customer because of the configuration problems that it causes for them. Without such modifications, the degree of integration possible will be severely limited.
- The initial cost of performing complex modifications to CASE tools is prohibitive, even if access is provided to source code. CASE tools are large systems, with many sized in the range of hundreds of thousands of lines of code. They are simply too large and complex to be modified by users.
- The real costs of CASE tool adoption must include significant funding for the transition process. If integration efforts effectively reduce the portion of the total CASE budget available for transition, then the CASE adoption effort is likely to fail. Some organizations that have attempted to provide their own

large-scale integrations of CASE tools have found that roughly half of the expense of the CASE initiative should be devoted to the transition effort involving awareness education, staff training, and proposal and operation support.

- The problems associated with maintaining a user-integrated CASE solution are significant. CASE tools change rapidly as new capabilities are added and new platforms are supported. As a result, users can expect frequent releases of tools. Obviously, if two or more CASE tools are involved in an integration, a new version of some tool involved in the integration can be expected even more frequently. The maintenance problem is compounded by the expected life span of the integrated tools. To maximize an organization's return on investment, the tool suite used to develop application software should be available throughout the life cycle of that software. Thus, customer provided tool suite integrations may have to be supported for decades.
- It is unlikely that customer integrations will be compatible with the framework support provided by commercial offerings, such as AD Cycle, AIX CASE, Cohesion, PCTE, and SoftBench. Such commercial framework offerings provide the best chance for integrated solutions.

In summary, simple integrations that require no changes to CASE tools, and that can be easily discarded for new technologies offered in the commercial sector, are probably realistic. Such integrations, which are often accomplished via scripts, have proven useful.

3.10 Can tools handle large applications?

Tools have made significant gains in their abilities to handle moderately sized applications. They now can operate on large numbers of data items, and in many cases perform adequately. This improvement is due both to the increasing sophistication of the tools, and to the enhanced performance of the computing platform on which they operate.

However, CASE tools continue to suffer performance degradation and inadequate capacity when used for the development and support of very large applications. Operations on a very large database or data dictionary can become prohibitively expensive due to the significant resources and time required.

In private discussions with the SEI, one builder of very large systems (hundreds of thousands, and millions of lines of code) indicated that no tool has yet been found for which some capacity limitation was not exceeded during system development. High quality tools tend to degrade gracefully and provide a work around at the system limits (often by allowing configuring of the system into multiple, smaller subsystems), while lesser quality tools degrade rapidly and fail catastrophically by losing data.

Users of CASE tools for very large systems have found the responsiveness of the tool vendor in problem situations to be critical to tool success. These users often express the sentiment that you can assume some problems with any tool for the largest systems, but a good working relationship with the vendor can lessen the impact of many problems.

3.11 How do I determine the "openness" of a tool?

Almost all vendors claim that their tools are "open" or "easy to integrate," yet it is obvious that these claims mean different things to different vendors. In spite of the difficulty in determining "openness," we can make a number of statements about an idealized open tool:

- The tool would adhere to the commonly accepted "open" standards, including UNIX (POSIX compliance), the X Window System for workstations, and the Motif look-and-feel.
- All features available at any level of the tool's user interface would be available via programmatic interfaces to all outside tools.
- The tool would be composed of separate and replaceable services. For example, configuration management capabilities would be provided by commercially available configuration management tools, rather than by the specific CASE tool. In addition, the configuration management tool would be easily replaceable by another, equivalent tool.
- The tool would provide a programmable user interface so that invocations of other external tools would be clean and consistent.
- The tool would be capable of notifying other tools about its actions as well as receiving information from other tools about their actions. In short, it must both "talk" and "listen."
- The tool would be capable of operating without being the focus of the user's attention.
- The tool would adhere to any commonly accepted data model.
- The tool would be available on a variety of common hardware platforms.

In reality, no available tool meets this set of criteria for openness. In fact, as tools continue to accumulate more services, they often become less open according to these criteria. We do, however, believe that tools tending toward the indicated architecture will be easier to integrate with other tools and more readily assimilated into developing integration frameworks. To determine whether a tool meets your current and long-term requirements for integration, a number of questions should be addressed concerning both your integration needs and the integration capabilities of the tools. Among the major questions an organization should address concerning the "openness" of a tool and potential for integration into the environment include:

- Is an integrating framework or backplane being used or planned for the future by your organization? By the tool vendor?
- What are the current capabilities of the tool in question with regard to integration frameworks?
- What use will the tool make of the integration framework? What changes will this entail for the tool? Is the level of granularity of the tool integration appropriate for your needs?
- What current and future standards are part of your strategy? How well does the tool meet your emerging standards?

- What interfaces would ideally be available to allow you to integrate the tool into your environment? What tool interfaces are available?
- What is the quality of the documentation describing the interfaces to the tool?
- What tool data is actually accessible outside the tool? How is it accessible?
- What contextual information is necessary for full interpretation of tool data?
- Is this contextual information available outside the tool? How?
- How much data is unique to the dictionary of the individual tool?
- Is there incompatibility between the dictionaries of the current tool and new tools?
- What is the granularity of the data that is shared?
- Can the tool invoke other tools? How?
- At what points in processing can the tool be invoked?
- Can the tool operate as a "slave" to another tool?
- Can the tool create predefined links to other tools? How well do these links fit your intended use?
- Can the tool create arbitrary links to other tools?
- Does the tool utilize a proprietary database, CM system, and/or user interface system, or can the tool be mated to a variety of mechanisms?
- Is the tool constructed as a monolithic entity, or as a set of semi-independent subsystems?
- What kind of process integration is available, either directly between the tool and other tools, or indirectly through a framework?

3.12 Must all tools be "open"?

Unless a tool is to be used in total isolation, and the work products of that tool are unrelated to other activities in the software process, then the tool should be "open". Of course, there are varying degrees of openness necessary for different tools to function effectively with other tools. For example, we might expect that an analysis and design tool would need a greater degree of openness than an electronic mail tool. By this, we imply that the analysis and design tool should provide us access at the semantic level; while with the mailer, we may only require the ability to invoke the mailer, receive mail, and decode the (ASCII text) form of the message. Simply put, we wish to know how and why the analysis and design tool performs an action, but we only need to know how the mailer performs an action (how it can be invoked).

[Brown & McDermid, 1991] define various levels of tool integration, including:

- The carrier level, which is represented by the sharing of byte streams or data file formats. Each tool is responsible for analysis of the shared information. Carrier level integration is demonstrated by shared file formats and byte streams of the UNIX operating system, and is commonly what we expect of mailers.

- The lexical level, which is identified by the sharing of data structure formats, often among a closely related family of tools. Documentation tools which share a common data format and operating conventions (such as the use of "." at the beginning of a line) use this approach to integration. Unfortunately, the conventions for understanding the structure continue to be embedded in the individual tool, and each input has to be analyzed for conformance to the convention.
- The syntactic level, in which the rules governing the formation of data structures are agreed on within the tool set. This level of integration is represented within CASE tool sets that share a common data structure.
- The semantic level, in which a common perception of data structure is combined with a common perception of the underlying semantics. Object-oriented and entity relationship databases are directly aimed at this level of integration.
- The method level of integration, in which agreement is reached not only on semantic issues, but also on the development process supported, and on the tools which support each part of the process. This level of integration is necessary to avoid overlaps and inconsistencies between tools.

Brown and McDermid further point out that the tighter, more complete the level of integration used in a tool set, the more difficult it is to reuse individual tools across applications or environments. As a result, we must make pragmatic decisions concerning the level of integration that is appropriate (and therefore, the level of openness necessary) for tools. It is suggested that within a life cycle phase, tighter forms of integration are appropriate (such as the semantic and method levels of integration of compilers and debuggers allowing them to operate on common data structures). Between life cycle phases, looser integration, perhaps at the semantic level, may be acceptable.

3.13 Will vendors customize a tool for me?

Conversations with vendors indicate that they frequently have hundreds of different customers requesting unique features. Often the customer assumes that others would also want the feature, and fails to see the uniqueness of the request. CASE tool vendors also are faced with a large backlog of features to be implemented that are not unique. They must carefully position resources to address the more universally desirable features. Finally, each customization of a tool represents another configuration of the tool to be maintained. Vendors complain heartily about the current number of tool configurations necessary to cover the market, even without considering custom configurations.

In light of these backlogs, vendors address tool customization in three ways: they will discourage customization of tools, they will provide a service (at significant cost) to customize the tool, or they will provide a degree of customization capability within the tool. Only for the very largest customers will a vendor customize a tool set at no cost.

Even if the vendor will customize a tool, it is unclear whether such customization is in the long-term interest of the customer. There is little assurance that the vendor will continue to support

a tool configuration with a relatively small customer base. The support that is offered can be expected to come after upgrades to the mainstream tool configuration. Such customization may also limit the degree to which the tool configuration will be included in integrations with other tools.

The customization features built into many tools represent a viable (if limited) option for tool customization. Such customization features normally are represented in the mainstream configuration of the tool, and are therefore adequately supported. In addition, features such as configurable menus can provide a mechanism for tool integration.

3.14 What do vendor integration claims mean?

Many vendors have begun to form strategic alliances with other vendors in order to ensure that their tools can interact. These integrations, termed "CASE coalitions" [Wallnau & Feiler, 1991], offer a strategic marketing advantage to the tool vendors, as well as a varying degree of integration to users.

In the near term, CASE coalitions may provide a practical solution to integration of tools into a cohesive tool set. However, the individual tools of such integrations continue to operate in an "egocentric" manner. Each maintains its own database and supports its own paradigms for software development. Each maintains a unique concept of multi-user support and configuration management. These non-integrated qualities of CASE coalitions will place a practical limit on the types of tools that can be included into the tool set, as well as the degree of integration provided.

A second set of problems occurs due to the proprietary nature of these coalitions. Vendor agreements between tools of CASE coalitions limit the possibility that other, competing tools may become members. Thus, the choice of the user is limited. In addition, the strength of a coalition offering may be put at risk by the relative weakness of any single tool in the coalition.

Finally, CASE coalition integrations are primarily ad hoc, in that vendors utilize the primitive integration interfaces that are available. As such, these integrations are difficult to tailor. In addition, it is unclear whether they can survive to compete against other developing frameworks. Thus, any purchase of coalition technology should be viewed as a short-term solution, and upgrade paths must be considered.

3.15 What do CASE tools cost?

A large number of factors can influence initial and continuing costs of developing and maintaining a state-of-the-art software development environment. Among these factors are the large number of tools necessary in such an environment, the costs for the initial installation, long-term maintenance costs, the costs of the supporting hardware and software environments, and the costs for additional and better trained staff. Some of the costs are associated specifically with those of a CASE environment. Others are connected to the maintenance of a sophisticated distributed computing environment.

3.15.1 Principal Cost Drivers

[Huff, 1992] identified a number of principal cost drivers for a CASE adoption budget. Some of these cost drivers include:

- the scope of the CASE adoption effort, including the number and type of projects for which the tool will be adopted,
- the complexity of the existing environment which will be affected by the CASE adoption effort,
- the size of the organization which will adopt the tool,
- the history of the organization in dealing with major changes to the culture,
- the current practices in the organization,
- the skill level of targeted end users, and
- the speed at which the transition is to occur.

3.15.2 Start-Up Costs

There is not substantial empirical data available on the cost of providing such a full set of software development tools. However, instructive lessons can be learned from the projected costs associated with the implementation of a tool and the associated support for a staff of 75 engineers. Obviously, these costs could vary substantially depending on the characteristics and needs of the organization.

Figure 3-1 contains a list of the projected start-up costs from [Huff, 1992]. Additional cost estimates are available from [Grochow, 1988].

Initial Investment	Total Cost	Per Seat	Comment
Technical Assessments	\$138,750	\$1,850	Process, standards, infrastructure, market, etc.
Organizational Assessments	\$22,500	\$300	Organization, people, economic
CASE Software	\$187,500	\$2,500	Estimated average price per package
Workstations	\$562,500	\$7,500	Estimated average price per workstation
Skills Development — Training	\$375,000	\$5,000	Twice the value of CASE software
Tool Consultants	\$50,000	\$667	50 days at \$1,000/day
Total Initial Investment:	\$1,336,250	\$17,817	

Adapted From: Grochow, J. M. "Justifying the Cost of CASE." *Computerworld* (February 8, 1988).

Figure 3-1 Start-Up CASE investment

According to the data presented in Figure 3-1, the total start-up costs associated with the investment in a sophisticated tool for 75 users would be approximately 1.3 million dollars. It

should be noted that this estimate includes \$562,500 allocated toward the purchase of a set of workstations at \$7,500 per unit. Such a purchase may or may not be necessary, depending on whether appropriate hardware is already available. In addition, if high performance workstations are necessary, costs may easily exceed \$7,500 per unit. It may be possible to reduce this cost by purchasing a smaller number of high performance workstations supported by X terminals.

3.15.3 Ongoing Costs

In addition to the start-up costs associated with the tool implementation, ongoing costs must be considered. An estimate of these costs is provided in Figure 3-2 [Huff, 1992]. Again, additional estimates are provided in [Grochow, 1988].

Ongoing Operations	Total Cost	Per Seat	Comment
Software Engineering Support Group	\$150,000	\$2,000	3 people at \$50,000 (salary & benefits)
Hardware Maintenance	\$37,500	\$500	Estimated \$500 per year for each workstation
Software Upgrades/Maintenance	\$28,125	\$375	Estimated 15% per year for each workstation
Ongoing Training	\$39,375	\$525	3-days' annual training per person at \$175/day
Conferences — User Group Meetings	\$12,000	\$160	2 people attending 4 meetings at \$1,500/meeting
Miscellaneous	\$5,000	\$67	Books, subscriptions, publications
Total Ongoing Costs	\$272,000	\$3,627	

Adapted From: Grochow, J. M. "Justifying the Cost of CASE," *Computerworld* (February 8, 1988).

Figure 3-2 Ongoing CASE Costs

It is possible that the costs for a Software Engineering Support Group may far exceed the suggested \$150,000 for three support personnel. Experience within the SEI indicates that the maintenance of network software, particularly for heterogeneous networks, can be demanding and requires both a greater number and a more highly trained (and paid) staff. Some experienced users suggest that the complexity and management of a sophisticated CASE analysis and design tool is more akin to that of an operating system than to that of a less sophisticated tool such as a simple text editor.

3.16 What determines the impact of a tool?

The potential impact of a tool on an organization is difficult to predict because many factors are involved. In our view, CASE impact depends on an interrelated network of factors (Figure 3-3).

Although much has been published about CASE, the early anecdotal studies of the impact of CASE tools did not distinguish among the effects of tools, associated changes to the organization's process and methods, or the way in which tools were adopted. Such factors can have major conflicting or confounding effects on productivity or quality. Additional research will be needed to separate these influences.

Figure 3-3 does suggest that tool characteristics, while important, represent only one of the factors that determine the effectiveness of tools. There is a need to carefully account for organization-specific factors, such as the size, resources, and culture. The way in which tools are adopted can have an additional long-term impact.

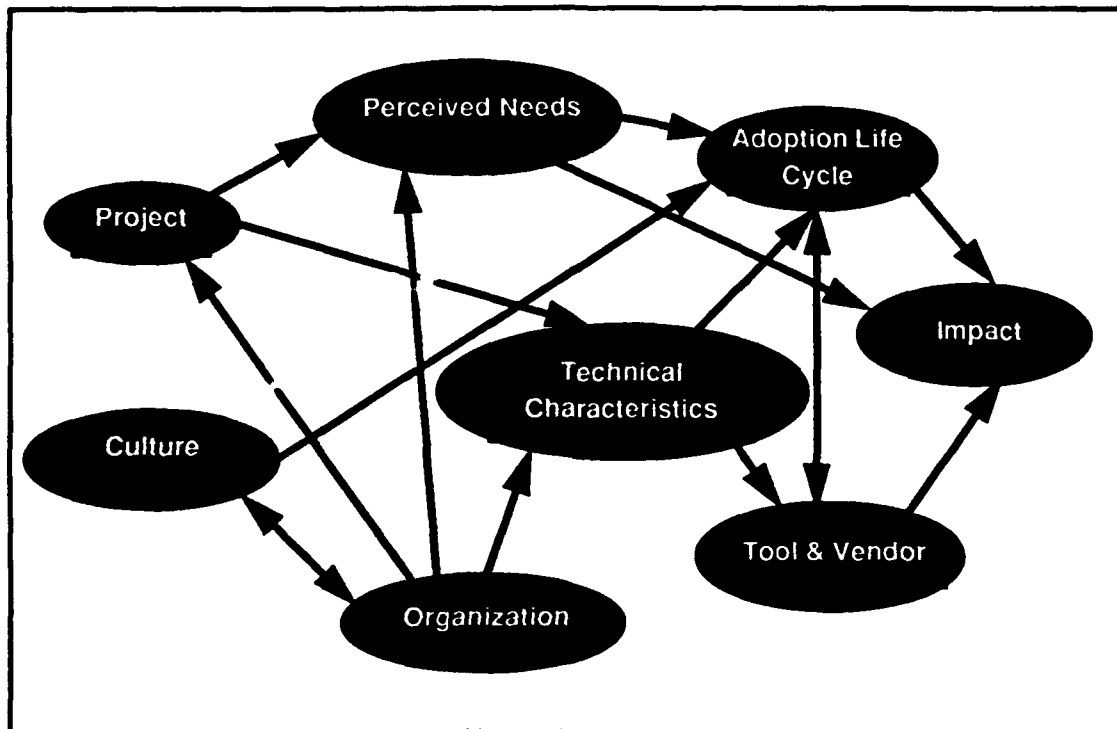


Figure 3-3 Factors That Influence CASE Impact

3.17 How do I maximize my return on investment?

Return on investment (ROI) depends on the development of a sound strategy that makes sense for an organization. As previously indicated, there are several prerequisites that must be satisfied. An organization should be serious about its software engineering process, and should be actively involved in an improvement effort. There should be a set of development methods in place. Given that the prerequisites are in place, there needs to be an understanding of the organization and what works within an organization, including issues of cultural change. A full range of tools, working together to solve a software engineering problem, may be more effective than any individual tool. The set of tools must be integrated within the context of an overall tool strategy. Standards and practices must be instantiated for a particular

organization, with specific concern for the process and methods of that organization. Cultural issues are then addressed, along with the roles and management checkpoints of the adoption life cycle.

3.18 What are major non-technical tool considerations?

Perhaps the most difficult hurdle to overcome is the idea that by introducing a CASE tool in the work environment, the implementor of the tool is actually changing the way that people do their work both physically and mentally. Change can often make people feel uncomfortable.

Although it might appear that during the adoption process the main concerns are the technical issues surrounding the selection of the CASE tool, many of the cultural aspects of the adopting organization can have significant impacts as well. Issues such as morale, perceived resistance to tools, and roles that people in the organization fulfill need to be considered as issues that affect the adoption process.

Morale is not something that immediately comes to mind when one thinks about the adoption process. Upon further inspection, it makes sense that an organization currently experiencing a low point would be least likely to accept new ideas. The prevailing apathy can grow quickly. Under these circumstances motivating such an organization to accept new technology can be a difficult task.

Perceived resistance is also a very real problem to overcome. Tools can enforce a methodology that is perceived as being restrictive. This is just one reason that software engineers give for not using CASE tools. However, getting software engineers to accept a new technology is no different than trying to get any other group of people to change the way they work. Studies have shown that although you can have limited success by mandating a change, the most successful changes take place when an individual participates in the change process.

Besides resistance, it is also important not to encourage unrealistic expectations. Management must develop a clear understanding of the expected impact of CASE technology, as well as a time frame during which benefits will become evident. This understanding must be communicated to tool users. In the past, some tool adoption efforts have been oversold as the solution to all of an organization's software development problems. This approach has resulted in inevitable frustrations and disappointments.

Much research has been devoted to the roles people play in their organization and how those roles can affect the adoption process. One role has been identified as being essential to the adoption process: the change advocate. This person's job is to sell the idea to management and obtain the resources and commitment to proceed with the adoption process. Without this person, the adoption cannot take place. The implementor of the tool must recognize himself as the change advocate. At times, he will be expected to be a sociologist, psychologist, and a software engineer.

Further reading on technology transition can be found in the following texts (see the "References" section for further information):

- Bouldin, B. *Agents of Change: Managing the Introduction of Automated Tools.*
- Pressman, R. S. *Making Software Engineering Happen: A Guide for Instituting the Technology.*
- Przybylinski, S. & Fowler, P. J. *Transferring Software Engineering Tool Technology.*
- Tornatzky, L. & Fleischer, M. *The Process of Technological Innovation.*

3.19 Should I attempt to build my own tools?

There have been several attempts to develop proprietary CASE tools. In most such attempts, the complexity and development costs were severely underestimated. Competent CASE tools cannot be built quickly or on a small budget. The competing, commercially available tools often have undergone numerous large-scale enhancements over a number of years.

In addition, it is important to reflect on the other costs that an organization must assume in developing its own tools. An organization must assume complete responsibility for maintenance of the tool. It must develop its own training programs and materials, and provide personnel to direct the training. It must perform all of its own integrations. It must incorporate (continually) all of the latest technological advances for tool integration. In short, it is best to leave CASE tool development to the CASE tool vendors.

There are, however, many opportunities for organizations to develop low-risk, inexpensive tools that provide support for the organization's process. In many ways, these tools can be of equal value to the more expensive CASE tools. The tradition for this class of in-house tools is well established: organizations have long built scripts to automate documentation, code generation, and testing functions. Perhaps one of the most useful efforts that an organization can make is to review its existing in-house support, identify holes in coverage, and build local, small-scale tools to provide support. While these tools must also be maintained, they are relatively cheap to build and support, and at worst case can be thrown away at little cost.

3.20 What new tool technology is on the horizon?

The primary improvements in tool technology will come from the increasing levels of integration between tools. Capabilities such as broadcast messaging (offered in HP SoftBench) and live links (offered by FrameMaker and Interleaf) will make new forms of integration possible. In addition, it is expected that framework technology will attract increasing attention as technical hurdles are cleared.

Individual tools will continue to improve incrementally. Most analysis and design tool vendors are actively incorporating simulation and modeling capabilities into their front-end tool set. In

addition, expert system technology may become more common in reverse engineering and application generation.

Tool offerings to address new design techniques, such as object-oriented approaches, are expected to increase. In addition, tightly coupled code implementation environments currently available in the C community should become more common in the Ada world.

New government initiatives to build CASE environments have the potential to speed the transition from current single-point tools to integrated tool sets. It is unclear, however, whether tool and environment technology is sufficiently advanced to support fully integrated environments.

4 Toward a CASE Adoption Strategy

After discussing the adoption life cycle and the issues surrounding the adoption process, it is obvious that there is no simple route to selecting and adopting the best set of CASE tools for a particular software project. In light of the ongoing problems experienced by organizations adopting CASE tools, then, it is hoped that use of a well-founded CASE adoption process, which includes careful analysis of the organization and the tools prior to actual tool purchase, will help maximize the return on investment for CASE tools. With CASE tool adoption, the less that is known about the adoption effort prior to the selection and during implementation, the more it will cost in time, money, and satisfaction once problems are discovered.

By viewing the adoption process in context and developing a tool strategy, the probability is higher for a successful effort. Although it is likely that organizations that use this or similar processes will incur a significant up-front expense, it is expected that the costs can be recouped by more appropriate tool choices and the less painful transition to follow. Even if an organization concludes that no CASE purchase is appropriate, it is possible that productivity or quality gains may be realized by careful introspection concerning process, personnel, and tools, leading to improvements in those areas. Therefore, in addition to formalizing the process of tool adoption, a useful strategy should provide:

- a mechanism for shared exploration of management and engineer goals,
- a vehicle for dissemination of organizational and tool capability,
- a focus of commitment for tool purchases, and
- a suitable climate for tool advocates and champions.

The proposed CASE adoption strategy, and its relationship to the stages of the CASE adoption process (outlined in Section 2), is defined as follows:

- **Awareness and Commitment**
 - Assess the organization. Prepare an organizational assessment report and requirements document that details the characteristics of personnel, current technology, process, projects, and politics, and identifies organizational needs.
- **Selection**
 - Assess available technology. Prepare a technology assessment report identifying the characteristics of the individual tools, tool vendors, and tool user experiences.
 - Assess the suitability of the technology to your organizational needs and select a tool if appropriate. Prepare an evaluation and selection report documenting the suitability of tools, and develop an action plan.
- **Trial (Implementation)**
 - Pilot the chosen tool. Prepare a pilot summary report detailing the experiences of the effort, along with a set of lessons learned and an implementation plan.

- **Implementation Strategy**

- Transition tool into general use. Prepare a list of lessons learned that will facilitate future adoption efforts. Prepare an institutionalization plan that outlines the expected culture changes and training requirements, and the need for project standards and effective measurement capabilities.

- **Routinization**

- Institutionalize tool use.

This proposed CASE adoption strategy is detailed in Figure 4-1 and in the following sections.

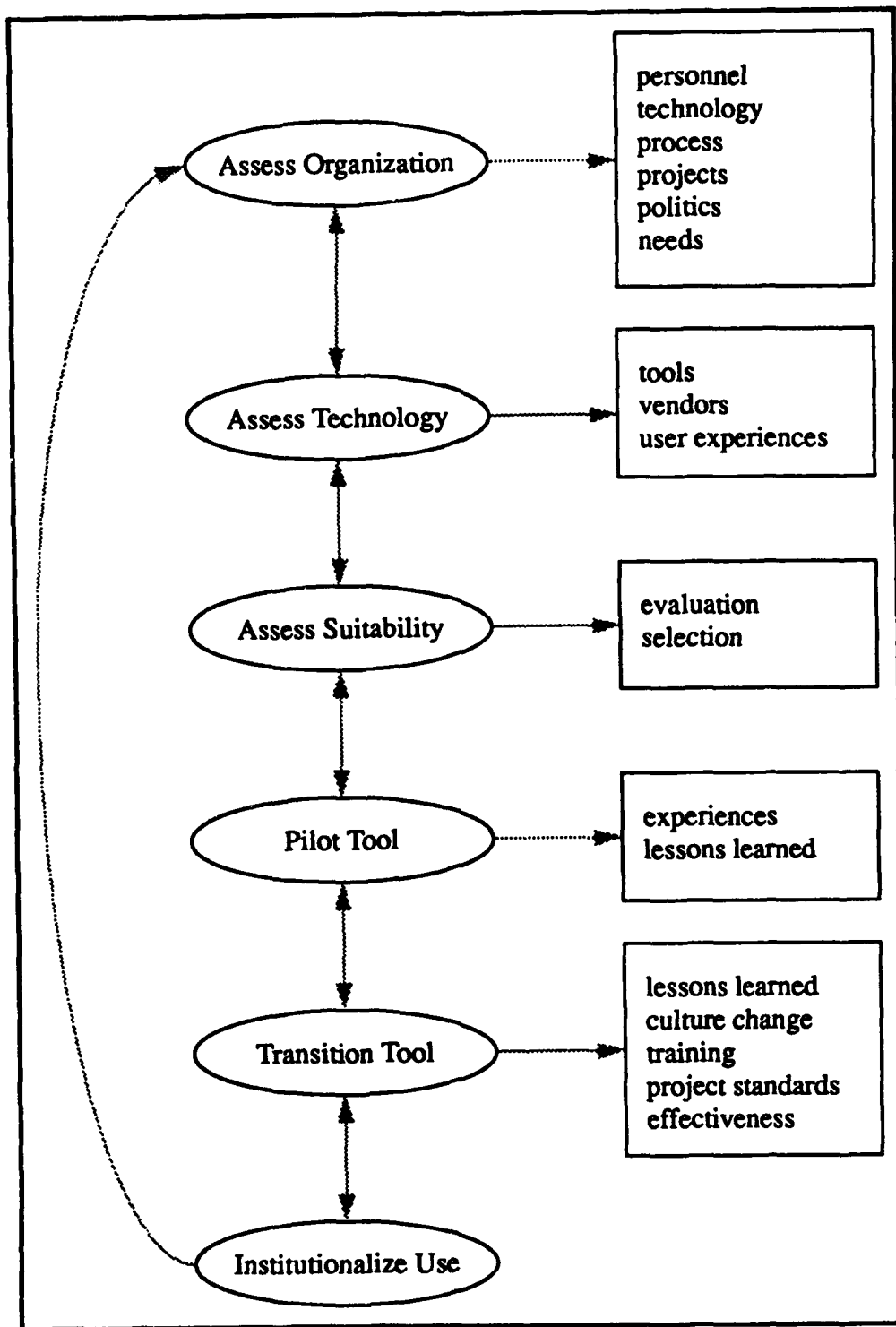


Figure 4-1 CASE Adoption Strategy

4.1 Assess the Organization

It is difficult to successfully adopt a CASE tool without paying careful attention to the needs, expectations, and abilities of the organization. The goal of the organizational assessment is to develop an organizational assessment report and requirements document that identifies the current state of the relevant personnel, process, and technology, identifies a set of improvements necessary within the organization, and anticipates procedural or cultural roadblocks to technology enhancement. Some of these improvements are likely to involve the purchase of software tools, while others may involve process or personnel improvement. The document should identify areas where software tool support is adequate and inadequate.

4.1.1 Evaluating the Organization's Personnel

A primary aim in evaluating the organization's personnel is to determine whether the majority of personnel are likely to view a change in a positive or neutral manner, or whether they will view change negatively, or actually sabotage change efforts. If your organization is currently unstable, it is more likely that change will be unsuccessful, since unstable organizations are often quicker to jump on a bandwagon of change, but are less likely to maintain change over the long term. If your organization is stable, initial change may be more difficult, but more likely to be sustained. With a more stable organization, you may also be more likely to anticipate problem areas and personnel, and attempt to minimize resistance. Among questions to be answered by an assessment of the organization's personnel are:

- How will employees (as a group and individually) react to the introduction of new tool technology? Is there a history of successful or unsuccessful changes that were attempted?
- Are there leaders, power centers, or brokers who will have a great impact on the manner in which new tool support is perceived?
- Is there a grass roots movement to encourage better tools and technologies?
- How much education will be necessary to orient users to the new tool?
- How stable is the staff? Is there a large turnover rate?
- Are there any specific hindrances or aids to communication in the environment? What are they?
- Are there any mechanisms and procedures in place for change, innovations, or suggestions?

4.1.2 Evaluating the Organization's Technology

The technological base of the organization includes not only the hardware resources on which software is developed, but also the languages, tools, techniques, and targets for the software. The organization's technology will heavily influence what tools are appropriate and available. The information gained from answering this set of questions can be used as part of a needs assessment which will direct software purchases. Questions to be answered include:

- What computing resources are available? What is the development platform? The development operating system?
- Are available resources always adequate to perform jobs on demand? If not, what resources are the bottleneck? How long is an average wait for the resource?
- What software tools are currently used in the organization? What other tools exist in the environment? What is the nature of those tools (e.g., commercial, home grown)?
- To what degree are tools used by software engineers integrated?
- What type and degree of networking is available to the development group?
- What programming languages are used?
- On the average, what percent of the code for a project is new? What percent is reused? What percent is maintained?
- What is the target platform for the project? Are there extreme constraints placed on the software due to the target platform? Is the target platform different from the development platform?

4.1.3 Evaluating the Organization's Process

While almost all organizations can benefit from some type of improved automation, organizations at different levels on the SEI capability maturity model (CMM) may have different tool needs. In addition, the quality of the organization's process will greatly impact the ability of that organization to incorporate new tools and to benefit from the tool ultimately. Questions to be answered concerning an organization's process include:

- How well defined is the process? How mature is the process from an organizational standpoint?
- Is the process, or portions of the process amenable to automation? What tools are available to enforce process and methods (e.g., bug trackers, CM tools)?
- What type of life cycle or development model is used by the organization (e.g., rapid prototyping, waterfall, spiral)?
- What is the organizational model (e.g., software factory, software pipeline)?
- What methods do you use (e.g., Gane and Sarson, State Charts, Entity Relationship)? How experienced is the organization with the method? What type of training was provided to staff in the method? Has the method been tailored or adapted to your specific organization?
- Are requirements analysis, specification and design, coding, and testing standards documented? Are they formal, or informal standards? What method is used to insure that standards are met?
- What metric information is gathered about the software process? How is it used? How long has it been gathered? What review processes are used?

- What types of documentation are produced during the software life cycle? What additional documentation capabilities are needed? Must a specific standard, like DOD-STD-2167A, be met?

4.1.4 Evaluating the Organization's Projects

While some organizations adopt technology at an organizational level, in many others technology adoption occurs at the project level. In fact, even when tools are chosen at the organizational level, transition must commonly occur at the project level. Questions concerning the organization's projects include:

- What is the average duration of a project in person months?
- How are your projects organized (e.g., dedicated staff, matrix organization, deep reporting structure, flat reporting structure)?
- What is the average size and range of ongoing software projects in terms of staff?
- What is the average size and range of projects in terms of source lines of code (SLOC) or function points (FP)? How do you measure SLOC/FP?
- Are there any special contractor or government security requirements that must be adhered to? How do they affect the project?
- Does a group exist to provide tool support to the project?

4.1.5 Evaluating the Organization's Politics and Perceived Needs

The dynamics of an organization necessitate a continual evaluation of the organization's infrastructure and capabilities, including the practices and techniques employed within the organization. Questions concerning the organization's politics and perceived needs include:

- How would your organization's productivity and quality compare to those of your competitors?
- What portions of the software life cycle are working best/worst? Are there specific portions of the life cycle that could be improved with new techniques?
- What additional documentation production capabilities are needed?
- Are inter-personal and group communications adequate? What additional facilities would make communication better?
- Are you currently collecting software metric data? What tool support are you using or do you expect to use to simplify collection of data?

4.2 Assess Available Technology

The goal of assessing the available technology is to develop a technology assessment report that accurately reflects the current state. This report should be far more than a listing of tools, but rather should reflect the expected benefits and costs of adopting the technology.

After determining that a tool purchase is appropriate for the organization, a careful analysis must be made of the tools available. The number of tools available may necessitate the generation of a two-step assessment: identification of the full range of technology, followed by careful analysis of a limited number of tools. A coarse-grained criteria for determining whether a tool is appropriate for further consideration may be used to determine which tools warrant more detailed analysis. For example, an organization concerned about the long-term viability of their tool provider may choose to grant full consideration only to tools with a relatively large market share. [Bouldin, 1988] suggests that detailed consideration should be limited to 3 or 4 tools.

There are a number of sources¹ of information about tools, including the vendors, and trade magazines such as *Computerworld*. It is difficult to develop a realistic impression of tools from these sources, however, since they have an interest in providing optimistic impressions of tool technology.

Regardless of the source of information about tools, (at least) three types of analysis should be performed. These include analysis of the tool, analysis of the vendor, and analysis of the experiences of other tool users.

4.2.1 Tool Analysis

As previously suggested, it may be necessary to perform a multistage analysis of tools to limit the number of serious contenders. Once identified, tools under serious consideration should be brought in house for an evaluation period. Many vendors will provide a trial version of the tool to be used during the selection phase, although some may request a refundable "deposit". Unwillingness to provide a trial version may be a warning flag indicating that the tool or vendor is deficient.

A number of tool rating scales or questionnaires are available for rating tools. While some of these scales provide a useful starting point for tool evaluation, others overemphasize the number of tool features and appearance of the user interface. As a result, some CASE vendors are working furiously to add new features that will attract potential customers using such "shopping lists". However, organizations already using CASE tools are often more concerned about plans to improve the usability of existing features rather than adding new features. In the end, it appears to be the quality of existing features rather than the quantity of what is promised that leads to successful tool use.

Generic tool rating scales also do not reflect the needs of the individual organization. However, based on the results of the organizational assessment, rating scales can be tailored to an organization or supplemented by use of tool criteria that have been carefully tailored for your or-

¹: A listing of CASE products can be found in the *CASE Outlook Guide to Products and Services* (published by the CASE Consulting Group, Inc., 11830 Kerr Parkway, Suite 315, Lake Oswego, OR 97035). More complete tool evaluations are available from Ovum Ltd. and the Software Technology Support Center at Hill Air Force Base (UT 84056). Overviews of relevant issues, including CASE adoption, can be found in Ed Yourdon's *American Programmer* magazine (161 West 86th St. NY, NY 10024-3411).

ganizational needs. Some of the questions particularly dependent on the unique qualities of the organization include:

- What are the most significant aspects (e.g., functions and features) of the tool that address the tool and process requirements of the organization? What are the strong and weak points?
- How does the tool's performance under load rate relative to the peak demand expected within the organization? How well does the tool support the degree of simultaneous use expected?
- How does the tool handle the scale of projects for which it will be used? Are mechanisms provided to decompose tool data into subsystems? Can data in the subsystems be related?
- How readily can the tool be adapted to the organization's environment, or to changing requirements?
- How compatible is the tool with existing methods, tools, processes, and environments?
- Is the overall product maturity at a sufficient level for the demands of the organization? Is the tool sufficiently robust?
- Is the tool update and delivery schedule appropriate for the organization?
- What is the expected time to proficiency (in months) for personnel to learn these tools?
- What is the expected cost (in terms of dollars and time) to acquire, implement, and maintain the tool (including acquisition costs for the tool and supporting hardware and software, ongoing maintenance costs, and education-training costs)? Will these costs jeopardize long-term commitment to the product?

4.2.2 Vendor Analysis

As indicated previously, under novel demands or when in support of large systems, even the best quality tool can suffer from flaws. The quality of the vendor has often determined the success or failure of a tool purchase. In addition, since to gain the expected benefit, a CASE tool must be in use for many years, it is particularly important that a CASE tool vendor be stable. Some of the questions to be considered in vendor analysis include:

- How long has the tool vendor been in the CASE tool market?
- Is the CASE tool business the company's primary business of the company?
- What are the vendor's annual sales in dollars?
- Do you feel that this vendor is a stable entity that will be around in 5 years?
- Does this vendor specialize in support for your software engineering domain?
- How would you rate the promptness and quality of their support staff?
- Does the vendor offer installation and or education/training? If so, how would you rate these services?

- How willing is the vendor to customize training to your specific needs?
- Does this vendor have a market reputation that you are aware of? If so, what is the nature of their reputation?
- Is the vendor's support staff located geographically close to your location?

4.2.3 User Experiences Analysis

The only way for an organization to determine which of the claims made by a vendor are based on fact, and which are hype is to attempt to gather information about the experiences of users of the tool. There are a number of places where such information is potentially available, including local user groups, computer bulletin boards devoted to software engineering, professional conferences, and personal contacts within other organizations. Among useful questions are:

- What issues are discussed at user group meetings?
- What is the general tenor of the discussion? Is it friendly, angry, neutral?
- What is the vendor's history of new releases? Are products released on or near the date announced? Are new product releases relatively robust?
- What quality and level of support have customers experienced?
- What war stories can current users share?
- What examples of the tool's use are available?
- What hints for making effective use of the tool are available?

4.3 Assess the Suitability of the Technology

The identification of available tool technology is necessary, but not sufficient to determine the suitability of a tool to your programming environment. All tools identified will have a set of strengths and weaknesses which will make the tool more or less suitable to your needs. In order to maximize your chances for success, you should compare the tool needs identified in the earlier tool requirements document with the idiosyncrasies of individual tools.

The end product of the selection process is a tool evaluation and selection report which identifies the candidate tool (or tools) and specifies which requirements the tool adequately addresses and which it does not. The report should also clearly define the expected impact of the tool over the short term (during the adoption process) as well as the expected benefit over the long term. An operational definition of expected impacts and benefits should be included to facilitate measurement of tool impact. An action plan should be developed to direct tool purchases.

4.3.1 Evaluating and Selecting the Tools

A major portion of the selection effort may be spent identifying and ranking the requirements (and criteria) for the tool based on prior analysis of organizational needs and available technology. The rationale for each of the requirements should be carefully recorded. In addition, a

method of assessing whether a tool meets the criteria should be defined. The requirements and associated rationale will become useful both in identifying the appropriateness of various tools and in evaluating the eventual effectiveness of the tool and the adoption process. Ideally, if a careful job was done in developing organizational and technology assessment documents, the creation of selection criteria should be a straightforward job.

A number of criteria for tool evaluation and selection have been published. [Firth, Mosely, Pethia, Roberts, & Wood, 1987] includes a more comprehensive set of issues to address in selecting tools. Appendix B provides additional pointers to some of the available listings of tools and evaluations of specific tools. In general, tools should be evaluated for capabilities in a number of both technical and non-technical areas, including:

- The compatibility of the tool with the capabilities of personnel. While requiring a new set of skills from personnel may not necessarily cause a tool to be ruled out, it will influence training plans and eventual costs. Groups tasked with providing new tool technology to an organization have found that training costs can consume up to half of the groups resources.
- The match between the tool and the organization's perceived needs. A tool that does not meet the organization's real needs is bound to be a disappointment. Often the customer holds the vendor responsible for such failures, when in some cases it is due to customer misunderstanding or unclear definition of needs.
- The features provided by the tool. Obviously, a tool that does not provide a critical feature will be less valuable than one that does, all else being equal. However, it is important not to let the primary selection criteria become a "feature count".
- The technology used in the tool. The tool technology will profoundly influence not only the degree to which the tool will interact with other tools, but also the number of years it will be usable. This is not meant to imply that only tools using the latest technology are worthy of consideration. In fact, tools that use well accepted and understood technology (for example, relational databases) may prove to be more stable than tools based on new but less proven technology.
- The degree of support provided for process and methods. Strong support does not equate with inflexible encoding of process and methods, however. The SEI has accumulated evidence from both commercial and government organizations that rigid encoding of process and methods is poorly accepted by the users.
- The degree of integration within and external to the tool. Tools with good internal integration will carry over a change in one portion of the tool (e.g., a data flow diagram) to another portion of the tool (e.g., an entity-relationship diagram). Tools with strong capabilities for external integration will use consensus standards, and provide mechanisms of accessing tool data, and for controlling the tool from other tools.

- The overall tool performance and support for cooperative processing. Many of the current generation of tools suffer from performance limitations when used in support of very large systems. For example, developers of one large environment effort have not found a single COTS tool for which a size limit could not be broached when developing large systems. High quality tools degrade gracefully and provide a "work around" at the system limits (e.g., allow reconfiguration of the system into multiple smaller subsystems).
- The quality of the documentation and customer support available. Types of vendor support which are offered include tool installation, problem hot lines, training sessions, and user's groups. While it is possible to evaluate tool documentation, it is much more difficult to evaluate customer support prior to using the tool on a large scale project. The evaluator must rely on impressions gathered from other tools users (ask for a list of customers and a contact at the customer site), the vendor's past track record, and general impressions of vendor quality.

4.4 Pilot the Tool

Many organizations transition tools to their employees by "decree." Leaders in the organization declare that a tool will be used for the first time on an important project, or will become the official organizational standard as of a specific date. This approach to transition has led to a number of failures as inadequately prepared staff struggle with a tool and ultimately relegate it to the shelf.

Before attempting to finalize a tool selection or to use a tool in a critical project, it is suggested that the tool first be used on a non-critical path pilot project. The project should be legitimate and be representative of the type of work for which the tool was purchased. It should be of sufficient size to provide valid lessons learned for larger projects. The project should be headed by an experienced individual recognized as a leader in the organization. This individual should be supportive of the tool effort, but should also be viewed as capable of providing impartial feedback on the effort.

4.4.1 Experiences and Lessons Learned

The goals of the pilot adoption are to identify and validate procedures and standards necessary for adoption of the tool by the more general organization. The set of reports generated during this phase may include a summary of the pilot effort, a set of lessons learned, and an implementation plan based on the pilot experiences. Among the issues to be addressed by a summary report on the pilot adoption effort include:

- the development of realistic expectations and schedules for full-scale tool implementation,
- the development of a set of tool experts and advocates ("tool champions") who can seed the organization,
- the documentation of the use of the tool and its role in the organization,

- the tailoring of the tool, the methods, and the organization's process for most effective use of the tool,
- the identification of training needs,
- the fostering of a "safe" atmosphere where individuals can learn about the tool without excessive pressures,
- the development of an organizational plan for full-scale implementation,
- the development of standards and guidelines for tool use,
- the development of a plan to address anticipated problems with the tool, the process, and the personnel, and
- the development of an implementation plan that identifies a strategy for tool implementation, starting with those projects most likely to accept the tool and to experience success, and gradually transitioning the technology to projects less ready to receive the tool technology.

4.5 Transition the Tool

Prior to moving toward full-scale implementation of the tool, the results of the pilot effort should be evaluated to determine whether further piloting is necessary. Assuming the results of the pilot suggest that full-scale implementation of the tool is possible, an implementation plan can be developed.

The tool can then be introduced to selected "live" projects based on the implementation plan. It is important to recognize that enforcing mandates across the organization at this point will often lead to failure. The history of transition efforts suggests that complete penetration in the organization can be slow, and loss of support at any stage can doom the new technology. The goals of the implementation effort, therefore, are to encourage tool adoption, maintain support, and disseminate the tool as organizations become ready. This can best be accomplished by acknowledging problems, developing a problem-solving climate for addressing those problems, and rewarding those individuals and groups that achieve success with the tool.

As in the pilot phase, one end product of the implementation phase should be a document containing a set of lessons learned from implementing the tool. Of particular importance are those insights that would facilitate future adoption efforts. A second product may be an institutionalization plan identifying ongoing support needed based on the lessons learned from adopting the tool.

4.5.1 Culture Change

In addition to containing a schedule, the implementation plan should also discuss the difficulty of changing the organization's culture. Cultural change within the organization has been compared to the grieving process, with periods of disbelief, anger, and (hopefully) acceptance of the new situation. In order to minimize the degree and duration of organizational disruption, [Implementation Management Associates, 1989] suggests that an organization must develop a systematic process:

- To define the transfer effort by ensuring that key players have a common view of why, what and how.
- To assess the organization's history of technology transition to identify major implementation barriers.
- To plan the management of change by identifying communication and reinforcement mechanisms.
- To identify key players and their roles in transition.
- To generate sponsorship for the transition effort.
- To manage organizational politics by assessing key players and identifying their potential sources of resistance, along with their perceived gain.
- To assess the inevitable resistance and identifying tactics to minimize it.
- To evaluate the support from the culture and planning to manage it.
- To assess the individuals facilitating the change and developing tactics to increase their skill and motivation.
- To plan, execute, and monitor the implementation.

4.5.2 Training

Hopefully, any training on the method itself has already been completed. Although not a preferred solution, it may be necessary to combine the method and tool training rather than waste the time to provide training separately. Assessing an individual's training needs at this point is an important task. Most people will not be instantly transformed from paper-and-pencil users to tool "gurus", but they will respond to relevant training. True commitment and expertise may come later, after improvements in productivity and quality due to tool use are evident.

[Bouldin, 1989] states that the selection of training will depend on many factors:

- the complexity of the product that you are developing,
- the quality of the courses that are commercially available,
- the amount of money that is available in your organization's budget,
- the time that your management is willing to allow for training,
- the experience level and interest of the users of the tool, and
- your own interest and ability in the area of teaching.

These factors should be weighed prior to selecting the proper training for your organization. The complexity of the product itself will probably have the greatest impact on the amount of training required. Your own expertise in this area should also not be discounted. The SEI has had significant success with programs to "train the trainer." Instead of paying for commercial courses for a large group, it is much more economical to train an in-house instructor and let him or her train their organization. This also allows the instructor to tailor the course to include particulars that are only relevant to their own organization.

4.5.3 Project Standards

Imposing standards can present some difficulties, but it is essential to the success of the project. Given the fact that almost everything is built by more than one person, it is important to determine how the tool is used, both individually and by groups. Establishing standards including naming conventions will enable a smoother transition between software life cycle phases. Keep in mind that existing standards should always be considered prior to adopting a new standard.

Below is a list of standards identified by [Smith, 1989]:

- Standards for using the tool.
- Naming conventions that will both be consistent with the methodological needs and will enable the use of the required analytical reports.
- Standards for backing up the database, together with standards for data sharing, and for locking and protecting the master copy of the database.
- Security standards.
- Report standards for each stage of the project's life cycle.
- Standards for monitoring the work of individual analysts and developers. A number of tools enable the development of reports on the work of each individual analyst over specific time periods.
- Standards and techniques for interfacing with other tools.
- Standards for documentation and document production.
- Quality assurance standards.

4.5.4 Evaluating the Effectiveness of CASE Tools

To determine how effectively a new CASE tool increases either productivity or quality, or both, the first step is to measure your current effectiveness. Unfortunately, few organizations currently employ an ongoing measurement and process improvement program.

If you are attempting to prove the effectiveness of CASE and its ability to improve productivity, a measurement system is required. The measurement must include, as a minimum, the ability to capture elapsed time and dedicated person hours, project complexity (possibly as measured by function points), and quality as measured by numbers of changes to design and code [Andrews, 1989].

Few metrics exist for determining the success of a CASE tool adoption effort. In fact, many experts believe that the real value of CASE tools lies in improved quality that leads to decreased maintenance costs. This type of benefit can be substantial over a period of time because maintenance often accounts for as much as 70% of the budget of a data processing organization. It is important to recognize that this potential impact will not be felt immediately.

Although a CASE tool can take as long as five years to reach its potential, it may develop incrementally over the short term. For this reason, regularly measuring performance gains can

aid evaluation. However, evaluation must start with a realistic measurement of the current environment prior to tool adoption, and must maintain consistent data gathering procedures over time.

4.6 Institutionalize Tool Use

Many tools which have been purchased and used for a period of time by the intended audience eventually fall into disfavor and are abandoned. This may be a symptom indicating that the tool never was accepted as part of the corporate culture, perhaps because the organization did not recognize the importance of *continued emphasis on the tool*. Even when a complex tool has moved into general use, it must be supported at a level far greater than more simple tools. Suggestions for continued support of a tool that may help routinize its use include:

- Continue support for ongoing training. Between new revisions of the tool and new staff, it is likely that training needs will continue indefinitely.
- Develop and implement policies for handling tool updates. Installation procedures and responsibilities must be clearly defined. Check-out procedures must be identified to determine whether a new version meets standards for quality, as should upgrade procedures to transform existing tool databases to new formats. Potentially, the configuration of tool versions and the supporting environment (other tools, the operating system, etc.) must be managed.
- Mechanisms should be established for internal sharing of experiences. *Potentially valuable devices include bulletin boards, news letters, and reuse libraries.*
- Mechanisms should be sought out for the sharing of experiences externally. These may include user groups, workshops, and published articles.
- The relationship with the vendor should be cultivated in order to stay abreast of plans and to insure that the vendor addresses feedback from your organization.
- Mechanisms to continually promote tool use should be developed. These may include *recognition for expert use and the establishment of a career path for individuals particularly interested in environments and tools.*
- Continual assessment of software quality and productivity is essential to identify that you are in fact improving, and to provide early notification if your strategy is failing.

5 Conclusion

It is likely that more than half of all CASE tools purchased are no longer in use. Besides the cost of the tool itself, this wasted expense may also include hardware costs, training costs, and lost employee productivity.

In spite of what some vendors will tell you about the ease of adopting their tool, no one can guarantee your success. To a great extent, both the quantity and quality of success will depend on how well your organization manages the adoption process.

There is no simple route to selecting and adopting the best set of CASE tools for your organization. However, by viewing CASE tool adoption in the larger context of organizational needs, and by developing a tool strategy as part of a larger organizational improvement strategy, you increase the probability for a successful effort.

A good analogy for the process of CASE tool adoption can be found in the software development process. Careful definition of requirements is essential. If an error is discovered during the requirements phase, it is relatively inexpensive to fix. The longer the error remains undetected, the more it will cost to correct. With CASE tool adoption, the less that is known about the adoption effort prior to the selection and during implementation, the more it will cost in time, money, and satisfaction once problems are discovered.

References

- Andrews, D. C. "CASE Users Find Implementation Trail Tough, But Following Correct Path Results in Gains." *MIS Week*, 10, 6 (Feb 1989), 48-50.
- Bouldin, B. *Agents of Change: Managing the Introduction of Automated Tools*. Englewood Cliffs, NJ: Yourdon Press, 1989.
- Brown, A. W. *Database Support for Software Engineering*. New York: Wiley, 1989.
- Brown, A. W. & McDermid, J. A. "On Integration and Reuse in a Software Development Environment." *Software Engineering Environments '91*, F. Long & M. Tedd (Editors), Ellis Horwood, Mar 7, 1991.
- Chappell, C., Downes, V., & Tully, T. "Real-Time CASE: The Integration Battle." *Ovum*, 1989.
- Feuche, M. "How to Use CASE Technology." *MIS Week*, 10, 37 (Sep 1989), 29.
- Firth, R., Mosley, V., Pethia, R., Roberts, L. & Wood, W. *A Guide To The Classification And Assessment Of Software Engineering Tools*. Software Engineering Institute Technical Report CMU/SEI-87-TR-10, ADA213968, (Aug 1987).
- Forte, G. "CASE: An Industry in Flux (Part 2)." *CASE Outlook*, 2, 4 (1988), 1-17.
- Fowler, P. & Rifkin, S. *Software Engineering Process Group Guide*. Software Engineering Institute Technical Report CMU/SEI-90-TR-24, ADAS235784, (Sep 1990).
- Grochow, J. M. "Cost Justifying CASE Requires Specific Identification." *MIS Week*, 9, 26 (Jun 1988), 35.
- Grochow, J. M. "Justifying the Cost of CASE." *Computerworld* (Feb 8, 1988), 19-20.
- Huff, C. C. "Elements of a Realistic CASE Tool Adoption Budget." *Communications of the ACM*, 35, 4 (Apr 1992), 45-54.
- Humphrey, W. *Managing the Software Process*. Reading, MA: Addison-Wesley, 1989.
- Implementation Management Associates (Additional materials developed by J. H. Maher). *Managing Technological Change*. Participants' guide in Software Engineering Institute Training Series, 1989.
- Jones, T. C. *Software Measurement and Estimation*. DCI Seminar Notes, Boston, MA, Aug 2-3, 1990.
- Necco, C., Tsai, N., & Holgeson, K. "Current Usage of CASE Software." *Journal of Systems Management*, 40, 5 (May 1989), 6-11.

- Paulk, M. C., Curtis, B., Chrissis, M. B., Averill, E. L., Bamberger, J., Kasse, T. C., Konrad, M., Perdue, J. R., Weber, C. V., & Withey, J. V. *Capability Maturity Model for Software*. Software Engineering Institute Technical Report CMU/SEI-91-TR-24, ADA240603, (Aug 1991).
- Pressman, R. S. *Making Software Engineering Happen: A Guide for Instituting the Technology*. Englewood Cliffs, NJ: Prentice Hall, 1988.
- Przybylinski, S. & Fowler, P. J. *Transferring Software Engineering Tool Technology*. Washington, DC: Computer Society Press of the IEEE, 1988.
- Przybylinski, S. M., Fowler, P. J., & Maher, J. H. "Software Technology Transition." Tutorial presented at the 13th International Conference on Software Engineering, Austin TX, May 12, 1991.
- Siegel, J., Stewman, S., Konda, K., Larkey, P., & Wagner, W. G. *National Software Capacity: Near Term Study*. Software Engineering Institute Technical Report CMU/SEI-90-TR-12, ADA226694, (May 1990).
- Smith, D. "Evaluating and Selecting CASE Tools." *Software Engineering: Tools, Techniques, and Practices*, 1, 3 (Sep/Oct 1990), 22-29.
- Smith, D. & Oman, P. "Software Tools in Context." *IEEE Software*, 7, 3 (May 1990), 14-19.
- Thomas, I. "PCTE Interfaces: Supporting Tools in Software Engineering Environments." *IEEE Software*, 6, 6 (Nov 1989), 15-23.
- Thomas, I. & Nejme, B. A. "Definitions of Tool Integration for Environments." *IEEE Software*, 9, 2 (Mar 1992), 29-35.
- Tornatzky, L. & Fleischer, M. *The Process of Technological Innovation*. Lexington: Lexington Books, 1990.
- Wallnau, K. & Feiler, P. *Tool Integration and Environment Architectures*. Software Engineering Institute Technical Report CMU/SEI-91-TR-11, ADA237810, (May 1991).
- Wasserman, A. "The Architecture of CASE Environments." *CASE Outlook*, 89, 2 (Mar/Apr 1989), 13-22.
- Wasserman, A. "Tool Integration in Software Engineering Environments." *Software Engineering Environments: Proceedings of the International Workshop on Environments*, F. Long (Editor), Springer-Verlag (1990), 137-149.
- Wood, D. & Wood, W. *Comparative Evaluations of Four Specification Methods for Real-Time Systems*. Software Engineering Institute Technical Report CMU/SEI-89-TR-36, ADA219187, (Dec 1989).

Appendix A Acronyms Used in This Guide

Ada	Department of Defense programming language
AIX	International Business Machine's Unix operating system
ASCII	American standard code for information exchange
CASE	computer-aided software engineering
CDIF	CASE data interchange format
CM	configuration management
CMM	Capability Maturity Model
COTS	Commercial off-the-shelf software
CPU	computer processing unit
DoD	Department of Defense
ECMA	European Computer Manufacturers' Association
FP	function points
MIS	management information systems
MOTIF	Open Software Foundation's graphical user interface
PCTE	portable common tool environment
POSIX	portable operating system interface
ROI	return on investment
SEI	Software Engineering Institute
SLOC	source line of code
STD	standard

Appendix B CASE Adoption Resources

The following tables provide a useful set of pointers to sources of information on Computer Aided Software Engineering (CASE) tools. This information, while not all-inclusive, does represent a significant cross section of the type of information that is available from commercial and government sectors.

Listed below is a summary of the following eight tables:

Table B-1: U.S. Government CASE Information Sources

Table B-2: CASE Industry Specific Reports/Directories

Table B-3: CASE Industry Specific Magazine-Based Buyer's Guides

Table B-4: General Software Industry Reports/Directories

Table B-5: Consulting Groups/Conferences

Table B-6: CASE Industry Newsletters

Table B-7: CASE Trade Shows

Table B-8: CASE User Groups

Table B-1: U.S. Government CASE Information Sources

Name	Contact/Source	Comments
GSA CASEbase	Judith Andrews GSA/OSDIT 5203 Leesburg Pike Suite 1108 Falls Church, VA 22041 (703) 756-4500	CASE database of vendors/ tools and government users/ evaluators
STSC CASE Database/ Toolbox PC	Air Force Software Technology Support Center Reuel Alder OO-ALC/TISAC Air Force Software Technology Support Center Hill AFB, UT 84056 AV 458-8045 (801) 777-8045	also contact for Joint Software Support Conference April 14-19, 1991 Salt Lake City, UT Sponsored by HQ USAF/SC and the Pentagon
Systems Engineering Tools for Computer-Aided Design of Ultra-Reliable Systems	Appendix A, Table A-10 CASE Tools BATTLE Tactical Technology Center 505 King Avenue Columbus, OH Sponsored by DARPA Available thru Defense Technical Information Center (202) 274-6847	Table of 173 CASE tools
Reviews of Selected System and Software Tools for Strategic Defense	Institute for Defense Analyses IDA Paper P-2177 Alexandria, VA Defense Technical Information Center Session Number ADA226 982 (703) 274-7633	Covers Software through Pictures, Teamwork, TAGS, Auto-G, DCDS, RDD, Statement, Refine, Design/ IDEF, 001, Foresight, Virtual Software Factory & Adagen
Evaluation of Existing CASE Tools for Tactical Embedded System	CECOM Center for Software Engineering US Army CECOM AMSEL-RD-SE-AST-SE Ft. Monmouth, NJ 07703 (908) 532-2342	Covers Teamwork, ProMod, EPOS, Software through Pictures, Statemate, Autocode, Model, CCC, Foresight, T & SuperCASE
Software Engineering TOOLS CATALOG	The Aerospace Corporation ATR-0089(8115)-1 El Segundo, CA 90245-4691	Covers Anatool, DataViews, Design Aid, Docwriter, Excelsior, FDM, Nexpert Object, PIES, P-NUT, PowerTools, Software Size Estimator, Software through Pictures, Statemate, Teamwork, TekCASE

Table B-2: CASE Industry Specific Reports/Directories

Source	Address/Phone	Title	Price
ACM SIGSOFT Software Engineering Notes vol 15 no1 Jan 1990 Page 79	Project SYTI Dept of Computer Science University of Jyväskylä Seminaarinkatu 15 SF-40100 Jyväskylä FINLAND	An Annotated CASE Bibliography	n/a
BIS CAP International	POB 68 Newtonville, MA 02160 (617) 893-9130	Implementing CASE: A Manager's Guide	\$595
CASE Consortium	Center for Study of Data Processing Washington University Campus Box 1141 Prince Hall 224 One Brooking Drive St. Louis, MO 63130-4899 (314) 889-4792	CASE Studies Annotated Software/Systems Bibliography (over 400 citations in 20 categories) CASE Studies Consortium MIS Industry Survey	unknown unknown
CASE Consulting Group, Inc.	11830 S.W. Kerr Parkway Suite 315 Lake Oswego, OR 97035	An Introduction to CASE: The Best of CASE OUTLOOK Annual CASE Directory The Executive's Guide to CASE	\$225 \$195 \$95
CASE Research	155 108th Ave. N.E. Suite 210 Bellevue, WA 98004 Note: CASE Research recently merged with Ernst & Young For more information contact: Ernst & Young's Center for Information Technology Strategy (617) 742-2500	"The Strategic Impact of CASE" - Volume I Video "The Strategic Impact of CASE" - Volume II Video Annual CASE Survey 1988 CASE Bulletins CASE in Practice reports Product Profiles The Second Annual Report on CASE	\$125 \$225 \$150 \$125 \$225 \$225 \$595
German National Research Center For Computer Science (GMD)	Western US Office 1942 University Avenue Berkeley, CA 94704 Publication	The CASE Products '90 (Macintosh HyperCard-based Public Domain Database) available on Internet, anonymous FTP sumex-aim.stanford.edu/info-mac/card/case-product-11.hqx	Free
Ovum Ltd	7 Rathbone Street London W1P 1AF England	Analysis Techniques for CASE: a Detailed Evaluation CASE Analyst Workbenches: a Detailed Evaluation CASE: the Next Steps Real-time CASE: the Integration Battle Reverse Engineering: Markets, Methods and Tools	\$995 \$995 \$995 \$995 \$1,850
P-Cube Corporation	915 Kings Canyon Road Brea, CA 92621 (714) 990-3169	CASEbase (a PC-based CASE database)	unknown
Forsette Systems	For information contact: Digital Consulting, Inc. 204 Andover Street Andover, MA 01810 (508) 470-3880	1990 CASE Evaluation Report	unknown
Software Productivity Group, Inc.	POB 294-MO Shrewbury, MA 01545-0294 (508) 842-4500	CASE Trends Industry Guide	\$179

Table B-3: CASE Industry Specific Magazine-Based Buyer's Guides

Source	Date	Page (s)	Title
Computer Decisions	1 Oct 88	p81(3)	<i>Change control meets CASE</i>
Computerworld	27 Mar 89	p77(5)	<i>CASE software products</i>
DEC User	1 May 89	p52(4)	<i>Vendors pack functionality into Case</i>
Digital Review	21 Nov 88	p61(7)	<i>CASE: tech toolkits for solid software.</i>
Digital Review	24 Jul 89	p37(7)	<i>Diverse CASE offerings deliver solid applications with speed and finesse</i>
Digital Review	23 Apr 90	p37(5)	<i>Project management packages offer sophisticated features</i>
Government Computer News	7 Aug 89	p56(4)	<i>CASE tools: timely assistance for PC-based software designers</i>
IEEE Software	1 May 90	p14(57)	<i>Tools Fair</i>
Macintosh Buyer's Guide	Fall 1989	p72	<i>Fall 1989 - Desktop Engineering Directory</i>
PC Week	21 Aug 89	p100(1)	<i>Education clearing the way for implementing CASE</i>
PC Week	21 Aug 89	p95(1)	<i>CASE spurs the re-engineering of users' hearts and minds</i>
PC Week	21 Aug 89	p98(1)	<i>CASE brings order to complex development efforts</i>
Software Magazine	1 Oct 90	p41(10)	<i>The race is on for tools enabled to IBM repository</i>
Software Magazine	1 Apr 89	p33(8)	<i>The CASE way of life: to each his own method</i>

Table B-4: General Software Industry Reports/Directorles

Source	Address/Phone	Title	Price
Data Decisions, Inc.	Cherry Hill, N.J.	<i>Data Decisions software</i>	unknown
DATA Sources	Ziff Communications Company One Park Avenue New York, NY 10016 (212) 503-5388	<i>DATA Sources</i>	\$495
Datapro	McGraw-Hill Info. Sys. Co. Computers & Comm. Info. Group 1805 Underwood Blvd. Delran, NJ 08075	<i>Datapro directory of microcomputer software</i> <i>PC Digest Ratings Report</i> <i>Software Digest Ratings Report</i> <i>Software Digest Macintosh Ratings Report</i>	\$779 unknown unknown unknown
NTIS	5285 Port Royal Rd. Springfield, VA 22161	<i>A directory of computer software</i>	unknown

Table B-5: Consulting Groups/Conferences

Name	Address/Phone	Conferences
Digital Consulting, Inc.	204 Andover Street Andover, MA 01810 (508) 470-3880	<i>Accelerating Applications Development (Using RAD, CASE...)</i> <i>Analyzing User Requirements</i> <i>Capers Jones: Software Measurement & Estimation</i> <i>CASE: The Next Generation</i> <i>Computer-Aided Software Engineering Symposium</i> <i>Data Modeling and CASE</i> <i>Evaluating CASE Tools</i> <i>IBM's AD/Cycle</i> <i>Implementing Software Engineering and CASE</i>
Extended Intelligence, Inc. (Associated with Dr. Carma McClure)	25 East Washington Street Suite 600 Chicago, IL 60602 (312) 346-7090	<i>CASE for the 1990s</i> <i>Re-Engineering, Repositories, Reusability</i>
Software Development Concepts (Associated with Dr. Paul Ward)	424 West End Avenue Suite 11E New York, NY 10024 (212) 362-1391	<i>The CASE/Real Time Curriculum</i>

Table B-6: CASE Industry Newsletters

Name	Source	Price
American Programmer	American Programmer 161 West 86th Street New York, NY 10024-3411	\$395/year
C/A/S/E Outlook Industry Report	CASE Consulting Group, Inc. 11830 S.W. Kerr Parkway Suite 315 Lake Oswego, OR 97035	\$395/year
CASE Strategies	Cutter information Group 1100 Massachusetts Avenue Arlington, MA 02174 (617) 648-8700	\$275/year
CASE Trends	Software Productivity Group, Inc. POB 294-MO Shrewsbury, MA 01545-0294 (508) 842-4500	\$49/year
CASE World News & Digest	Digital Consulting, Inc. 204 Andover Street Andover, MA 01810 (508) 470-3880	Free
Software Engineering Tools, Techniques, Practice	Auerback Publishers 210 South Street Boston, MA 02111 (800) 950-1216	\$145/year

Table B-7: CASE Trade Shows

Name	Sponsor/Contact	Comments
CASE World	Digital Consulting, Inc. 204 Andover Street Andover, MA 01810 (508) 470-3880	upcoming Los Angeles, CA March 5-7 1991
CASExpo	CASExpo Suite 1210 5203 Leesburg Pike Falls Church, VA 22041-3401 (703) 284-7330	
Tri-Ada	Daniel & O'Keefe Associates, Inc Conference Management 75 Union Avenue Sudbury, MA 01776 (1-800-833-7751)	
CASELab '90	Research & Technology Institute 301 West Fulton, Suite 718 Grand Rapids, MI 49504 (616) 771-6626	

Table B-8: CASE User Groups

Name	For information contact
International CASE User's Group	Computer & Engineering Consultants, Ltd. 18620 West Ten Mile Road Southfield, MI 48075-2667 Sponsored by CASE Research
CASE User's Network	Digital Consulting, Inc. 204 Andover Street Andover, MA 01810 (508) 470-3880 Sponsored by Digital Consulting, Inc.

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified			1b. RESTRICTIVE MARKINGS None		
2a. SECURITY CLASSIFICATION AUTHORITY N/A			3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release Distribution Unlimited		
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A					
4. PERFORMING ORGANIZATION REPORT NUMBER(S) CMU/SEI-92-TR-15			5. MONITORING ORGANIZATION REPORT NUMBER(S) ESC-TR-92-015		
6a. NAME OF PERFORMING ORGANIZATION Software Engineering Institute		6b. OFFICE SYMBOL (if applicable) SEI	7a. NAME OF MONITORING ORGANIZATION SEI Joint Program Office		
6c. ADDRESS (city, state, and zip code) Carnegie Mellon University Pittsburgh PA 15213			7b. ADDRESS (city, state, and zip code) ESC/AVS Hanscom Air Force Base, MA 01731		
8a. NAME OFFUNDING/SPONSORING ORGANIZATION SEI Joint Program Office		8b. OFFICE SYMBOL (if applicable) ESC/AVS	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F1962890C0003		
8c. ADDRESS (city, state, and zip code) Carnegie Mellon University Pittsburgh PA 15213			10. SOURCE OF FUNDING NOS.		
			PROGRAM ELEMENT NO 63756E	PROJECT NO. N/A	TASK NO N/A
11. TITLE (Include Security Classification) Guide to CASE Adoption			WORK UNIT NO. N/A		
12. PERSONAL AUTHOR(S) Kimberly Stepien Oakes, Dennis Smith, Ed Morris					
13a. TYPE OF REPORT Final		13b. TIME COVERED FROM TO		14. DATE OF REPORT (year, month, day) November 1992	
				15. PAGE COUNT 58	
16. SUPPLEMENTARY NOTATION					
17. COSATI CODES			18. SUBJECT TERMS (continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB. GR.	computer-aided software engineering, software engineering, CASE adoption, CASE technology		
19. ABSTRACT (continue on reverse if necessary and identify by block number) In an attempt to address the productivity and quality problems afflicting the software industry, many organizations are turning toward computer-aided software engineering (CASE) technology as a potential solution. Unfortunately, the inflated claims of vendors and unreasonable expectations of new users have led to many failed CASE adoption efforts. This guide answers questions organizations may have concerning CASE technology, and provides a strategy for the adoption of CASE tools into an organization. <div style="text-align: right;">(please turn over)</div>					
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED <input checked="" type="checkbox"/> SAME AS RPT <input type="checkbox"/> DTIC USERS <input checked="" type="checkbox"/>			21. ABSTRACT SECURITY CLASSIFICATION Unclassified, Unlimited Distribution		
22a. NAME OF RESPONSIBLE INDIVIDUAL Tom Miller, Lt Col, USAF			22b. TELEPHONE NUMBER (include area code) (412) 268-7631		22c. OFFICE SYMBOL ESC/AVS (SEI)

ABSTRACT — continued from page one, block 19